Verification, Validation and Evaluation of the Virtual Human Markup Language (VHML)

Examensarbete utfört i datavetenskap
av

Camilla Gustavsson
Linda Strindlund
Emma Wiknertz

LiTH-ISY-EX-3188-2002

2002-01-31

Verification, Validation and Evaluation of the Virtual Human Markup Language (VHML)

Thesis work performed in Computer Science
at Linköpings Tekniska Högskola
by
Camilla Gustavsson
Linda Strindlund
Emma Wiknertz

Reg no: LiTH-ISY-EX-3188-2002

Supervisors: Andrew Marriott and Don Reid, Curtin University of Technology
Examiner: Robert Forchheimer, Linköpings Tekniska Högskola
Linköping 2002-01-31

Avdelning, Institution
Division, Department

Institutionen för Systemteknik
581 83 LINKÖPING

Datum
Date
2002-01-31

Språk
Language
Svenska/Swedish
X Engelska/English

Rapporttyp
Report category
Licentiatavhandling
X Examensarbete
C-uppsats
D-uppsats
Övrig rapport

ISBN

ISRN
LITH-ISY-EX-3188-2002

Serietitel och serienummer **ISSN**
Title of series, numbering

URL för elektronisk version
http://www.ep.liu.se/exjobb/isy/2002/3188/

Titel
Title

Verifiering, validering och utvärdering av Virtual Human Markup Language (VHML)

Verification, Validation and Evaluation of the Virtual Human Markup Language (VHML)

Författare
Author

Camilla Gustavsson Linda Strindlund Emma Wiknertz

Sammanfattning
Abstract

Human communication is inherently multimodal. The information conveyed through body language, facial expression, gaze, intonation, speaking style etc. are all important components of everyday communication. An issue within computer science concerns how to provide multimodal agent based systems. Those are systems that interact with users through several channels. These systems can include Virtual Humans. A Virtual Human might for example be a complete creature, i.e. a creature with a whole body including head, arms, legs etc. but it might also be a creature with only a head, a Talking Head. The aim of the Virtual Human Markup Language (VHML) is to control Virtual Humans regarding speech, facial animation, facial gestures and body animation. These parts have previously been implemented and investigated separately, but VHML aims to combine them. In this thesis VHML is verified, validated and evaluated in order to reach that aim and thus VHML is made more solid, homogenous and complete. Further, a Virtual Human has to communicate with the user and even though VHML supports a number of other ways of communication, an important communication channel is speech. The Virtual Human

has to be able to interact with the user, therefore a dialogue between the user and the Virtual Human has to be created. These dialogues tend to expand tremendously, hence the Dialogue Management Tool (DMT) was developed. Having a tool makes it easier for programmers to create and maintain dialogues for the interaction. Finally, in order to demonstrate the work done in this thesis a Talking Head application, The Mystery at West Bay Hospital, has been developed and evaluated. This has shown the usefulness of the DMT when creating dialogues. The work that has been accomplished within this project has contributed to simplify the development of Talking Head applications.

Nyckelord
Keyword
Talking Head, Virtual Human, Dialogue Management, XML, VHML, Facial Animation, Computer Science, Human Computer Interaction

Abstract

Human communication is inherently multimodal. The information conveyed through body language, facial expression, gaze, intonation, speaking style etc. are all important components of everyday communication. An issue within computer science concerns how to provide multimodal agent based systems. Those are systems that interact with users through several channels. These systems can include *Virtual Humans*. A Virtual Human might for example be a complete creature, i.e. a creature with a whole body including head, arms, legs etc. but it might also be a creature with only a head, a *Talking Head*.

The aim of the *Virtual Human Markup Language* (VHML) is to control Virtual Humans regarding speech, facial animation, facial gestures and body animation. These parts have previously been implemented and investigated separately, but VHML aims to combine them. In this thesis VHML is verified, validated and evaluated in order to reach that aim and thus VHML is made more solid, homogenous and complete.

Further, a Virtual Human has to communicate with the user and even though VHML supports a number of other ways of communication, an important communication channel is speech. The Virtual Human has to be able to interact with the user, therefore a dialogue between the user and the Virtual Human has to be created. These dialogues tend to expand tremendously, hence the *Dialogue Management Tool* (DMT) was developed. Having a tool makes it easier for programmers to create and maintain dialogues for the interaction.

Finally, in order to demonstrate the work done in this thesis a Talking Head application, *The Mystery at West Bay Hospital*, has been developed and evaluated. This has shown the usefulness of the DMT when creating dialogues.

The work that has been accomplished within this project has contributed to simplify the development of Talking Head applications.

Acknowledgements

We would like to thank a number of people for helping us complete our Master thesis.

First of all, we would like to show our appreciation to the School of Computing at Curtin University of Technology in Perth, Australia, for their kindness and their hospitality to us as research students for one semester.

We would also like to thank Andrew Marriott, our supervisor during our 19 weeks at Curtin, who has put a lot of effort in supporting us and guiding us through our work. Without him, the project would have been less interesting and a lot harder. We would also like to express our thanks to his family, who invited us to their home and offered us help and support to find and equip a house for our stay in Australia.

Further, we would like to thank Simon Beard, at Curtin, for his opinions during the development of DMTL and DMT and for his engagement in creating Talking Heads from our pictures.

We are also grateful to Don Reid, our second supervisor at Curtin, for his direction and excellent teaching in the English language. Without him, our thesis would have been provided with many more grammatical mistakes.

We would also like to express thanks to our examiner Robert Forchheimer, at Linköping University.

Moreover, we thank Jörgen Ahlberg, at Linköping University, for giving us an introduction to MPEG-4 and his feedback on our first proposal draft.

We are also grateful to the members of the Interface group at Curtin, for feedback on *The Mystery at West Bay Hospital* and VHML.

We thank Hanadi Haddad for testing and commenting the dialogue in *The Mystery at West Bay Hospital*.

We would also like to express gratitude to Igor Pandzic, Mario Gutierrez, Sumedha Kshirsagar and Jacques Toen, who are members of the European Union 5th Framework, for their comments during the evaluation of VHML.

Also thanks to Ania Wojdel and Michele Cannella for their contribution with opinions about and proposed solutions to the structure of VHML.

We thank Michael Ricketts, for his technical support and excellent photography for our pictures for the Talking Head application.

We would also like to thank our opponents at Linköping University for excellent feedback, Erik Bertilson, Knut Nordin and Kristian Nilsson.

Finally, we thank Jonas Svanberg, Linköping University for technical support during preparations for the presentation in Linköping.

Camilla Gustavsson
Linda Strindlund
Emma Wiknertz

Linköping, 31 January 2002

Table of Contents

1 INTRODUCTION....17

1.1 AIMS....17
1.2 SIGNIFICANCE....18
1.3 PROBLEM FORMULATION....19
1.4 LIMITATIONS....19
1.5 METHODOLOGY....19
1.5.1 VHML....19
1.5.2 DMT....20
1.5.3 Demonstration and evaluation....20

2 LITERATURE REVIEW....21

2.1 TALKING HEAD INTERFACES....21
2.1.1 Applications....22
2.2 FACIAL ANIMATION....24
2.2.1 Reflections....26
2.3 FACIAL GESTURES....27
2.3.1 Facial expression....28
2.3.2 Facial parts....29
2.3.3 Synchronism....30
2.4 MPEG-4....31
2.4.1 Feature Points....31
2.4.2 Facial Animation Parameters....31
2.4.3 Neutral face....33
2.4.4 Facial Animation Parameter Units....33
2.4.5 Facial Definition Parameters....34
2.5 HUMAN SPEECH....34
2.6 XML....36
2.6.1 The XML document....37
2.6.2 Well formness, validation, DTD and XML Schema....38
2.6.3 XSL Stylesheet....39
2.6.4 DOM and SAX....39
2.6.5 XML Namespaces....39
2.7 VHML....41
2.7.1 EML....42
2.7.2 SML....42
2.7.3 FAML....43
2.7.4 HTML....44
2.7.5 BAML....44
2.7.6 DMML....45
2.8 DIALOGUE MANAGEMENT....45

3 VIRTUAL HUMAN MARKUP LANGUAGE....47

3.1 CRITERIA FOR A STABLE MARKUP LANGUAGE....47
3.2 GENERAL ISSUES....47
3.3 THE TOP LEVEL ELEMENTS....50
3.4 EMOTION MARKUP LANGUAGE....52
3.5 GESTURE MARKUP LANGUAGE....55
3.6 FACIAL ANIMATION MARKUP LANGUAGE....56
3.7 SPEECH MARKUP LANGUAGE....58
3.8 BODY ANIMATION MARKUP LANGUAGE....59
3.9 EXTENSIBLE HYPERTEXT MARKUP LANGUAGE....60

3.10 DIALOGUE MANAGER MARKUP LANGUAGE 60
3.11 DISCUSSION 60

4 DIALOGUE MANAGEMENT TOOL 63

4.1 DIALOGUE MANAGEMENT TOOL LANGUAGE 63
4.1.1 Dialogue 64
4.1.2 Macros 64
4.1.3 Defaulttopic 65
4.1.4 Topic 65
4.1.5 Subtopic 65
4.1.6 State 66
4.1.7 Stimulus 67
4.1.8 Response 67
4.1.9 Prestate, nextstate and signal 68
4.1.10 Evaluate 69
4.1.11 Other 69
4.1.12 DMTL example 69
4.2 REQUIREMENTS 71
4.2.1 Open file 71
4.2.2 Save file 71
4.2.3 Import file 71
4.2.4 Export file 71
4.2.5 Print file 72
4.2.6 Quit DMT 72
4.2.7 Edit 72
4.2.8 View 73
4.2.9 Options 73
4.2.10 Help 73
4.3 IMPLEMENTATION 74
4.3.1 DOM tree 74
4.3.2 The Graphical User Interface 74
4.4 PROBLEMS 75
4.4.1 Fully qualified names 75
4.4.2 XML-based 76
4.4.3 Print to file 77
4.5 TESTING 77
4.6 HOW TO USE THE SYSTEM 78
4.7 DISCUSSION 78

5 TALKING HEAD APPLICATION 81

5.1 INITIAL EVALUATION 81
5.1.1 Preparation 81
5.1.2 Discussion 82
5.1.3 Conclusions 83
5.1.4 Outcome 83
5.2 APPLICATIONS 84
5.3 THE MYSTERY AT WEST BAY HOSPITAL 85
5.3.1 Background 85
5.3.2 Design ideas 86
5.3.3 GUI 86
5.3.4 Creating the dialogue 88
5.3.5 A dialogue example 89
5.3.6 Structure 90
5.4 DISCUSSION 91

6 EVALUATION 93

6.1 VHML 93
6.1.1 Result *93*
6.1.2 Discussion *95*
6.1.3 Conclusions *96*
6.2 DMT 96
6.2.1 Discussion *97*
6.2.2 Conclusions *98*
6.2.3 Talking Head workshop *99*
6.3 THE MYSTERY AT WEST BAY HOSPITAL 100
6.3.1 Result *100*
6.3.2 Discussion *102*
6.3.3 Conclusions *103*

7 SUMMARY 105

7.1 FUTURE WORK 106
7.1.1 VHML *106*
7.1.2 DMT *106*
7.1.3 The Mystery at West Bay Hospital *107*

BIBLIOGRAPHY 109

GLOSSARY 115

INDEX 119

APPENDIX A: VHML WORKING DRAFT V. 0.4 129

APPENDIX B: DIALOGIE MANAGEMENT TOOL 181

APPENDIX C: VHML DTD 189

APPENDIX D: DMTL DTD 201

APPENDIX E: USER MANUAL 207

APPENDIX F: TEST SCHEDULE 225

APPENDIX G: THE MYSTERY AT WEST BAY HOSPITAL 229

APPENDIX H: VHML QUESTIONNAIRE 233

APPENDIX I: MYSTERY QUESTIONNAIRE 241

List of Figures

Figure 1. The *Olga*-character 22
Figure 2. The talking agent *August* and the 19th century Swedish author August Strindberg 23
Figure 3. *Ananova* 24
Figure 4. *Dr. Sid* in *Final Fantasy* 26
Figure 5. An emotion divided in the three parameters 30
Figure 6. FPs on the tongue and the mouth 31
Figure 7. The six different emotions used in MPEG-4 32
Figure 8. A model showing the FAPUs 33
Figure 9. A simple XML document 37
Figure 10. Blending namespaces. 40
Figure 11. Qualified names 40
Figure 12. A default namespace 40
Figure 13. A simple VHML fragment 41
Figure 14. A diagram of the greeting example 46
Figure 15. An example on how the transform function works from Swedish to English 49
Figure 16. The structure of VHML 49
Figure 17. An example of a VHML document, only using the top level elements 52
Figure 18. An example of a VHML document using emotion elements 54
Figure 19. An example of a VHML document using gesture elements 55
Figure 20. An example of a VHML document using facial animation elements 58
Figure 21. An example of a VHML document using speech elements 59
Figure 22. An example of a VHML document using the XHTML element 60
Figure 23. The structure of DMTL 63
Figure 24. The DMT GUI 75
Figure 25. *The Mystery at West Bay Hospital* GUI 87
Figure 26. The underlying structure of *The Mystery at West Bay Hospital* 90

List of Tables

Table 1. FAP groups 32
Table 2. Description of the emotions 32
Table 3. Description of the FAPUs 33
Table 4. Summary of human vocal emotion effects 35
Table 5. Standard entities in XML 37
Table 6. Elements in VHML 41
Table 7. A summary and description of the top level elements 50
Table 8. A summary and description of the emotion elements 52
Table 9. A comparison between nouns and adjectives for the emotion names 54
Table 10. A summary and description of the GML elements 55
Table 11. A summary and description of the FAML elements 56
Table 12. A summary and description of the SML elements 58
Table 13. A summary and description of the XHTML element 60
Table 14. DMTL elements 64
Table 15. Summary of the test results 77
Table 16. Information from the logged files 102

1 Introduction

Human communication is inherently multimodal. The information conveyed through body language, facial expression, gaze, intonation, speaking style etc. are all important components of everyday communication (Beskow, 1997). An issue within computer science concerns how to provide multimodal agent based systems. Those are systems that interact with users through several channels. These systems often include *Virtual Humans*, (VHs). A VH might, for example, be a complete creature, i.e. a creature with a whole body including head, arms, legs etc., but it might also be a creature with only a head. When a head is used as a user interface giving users information etc., the interface is described as a *Talking Head*, (TH).

The European Union 5th Framework Research and Technology Project, called *InterFace*, covers research, technological development and demonstration activities. It defines new models and implements advanced tools for audio-video analysis, synthesis and representation in order to provide essential technologies for the implementation of large-scale virtual and augmented environments. The metaphor, which inspires the project approach, is oriented to make man-machine interaction as natural as possible, based on everyday human communication means like speech, facial expressions and body gestures from the user as well as the VH (InterFace, 2001).

This Master thesis project was carried out in cooperation with the Department of Electrical Engineering at Linköping University, Sweden and the School of Computing at Curtin University of Technology, Perth, Australia. Both universities are part of the InterFace project.

The *Virtual Human Markup Language* (VHML) is being developed by the Interface group at Curtin (VHML, 2001). VHML is a markup language that will be used for controlling VHs regarding speech, facial animation, facial gestures and body animation. VHML is also a part of the InterFace project.

1.1 Aims

The main aim of this Master thesis project is to simplify the development of interactive TH applications. In order to do this, the project involves verification, validation and evaluation of VHML and thus making it more solid, homogenous and complete. Further, the aims of the project involve creating a tool, the *Dialogue Management Tool* (DMT), for constructing dialogues for TH applications.

The research aims to expand upon the work in the TH area done by Stallo (2000) in his honours work on adding emotion to speech, and by Huynh (2000) in his honours work on facial expressions. Reaching the aim will involve research into many different areas;

- *TH applications.* To get an overview of the existing applications and the advantages and disadvantages of using THs in user interfaces.
- *Facial animation.* To understand the importance of animating the TH in order to develop an effective user interface.
- *Facial gestures.* To understand the importance of facial expressions in order to get a natural TH.

- *Human speech.* To understand the importance of implementing emotions in the TH speech in order to develop an appreciated user interface.
- *MPEG-4.* To understand how facial animation of a TH is being accomplished.
- *XML*. To get an overview of the advantages and disadvantages of using XML as a base for a markup language.
- *VHML*. To get an overview of what the objectives are for VHML and what has been done so far.
- *Dialogue management.* To get an understanding of why dialogues are important concerning interactivity between a user and a TH as well as how a tool for creating dialogues can be useful.

The result of the project will be a new version of the VHML working draft, a dialogue management tool (the DMT), and two separate interactive TH applications. The applications aim to show the advantages of using the DMT, when constructing dialogues for an interactive TH, as well as demonstrate the functionality of VHML.

1.2 Significance

Simplifying the development of interactive TH applications is an interesting research issue, since the use of THs within the human computer interaction area currently has a high profile. Examples of applications using THs can be seen in section 2.1.1.

At present, different languages are used for developing different parts of the TH. For example, *Facial Animation Markup Language* (FAML), developed by Huynh (2000), can be used for facial animation and regarding speech there are, for example, *Speech Markup Language* (SML), developed by Stallo (2000), and *Synthesis Speech Markup Language* (SSML), developed by *World Wide Web Consortium* (W3C, 2001). These languages have been developed independently of each other. Using several different languages, which are not really connected and do not follow any standard, makes the development of TH applications harder than it would have been if the languages had been designed within the same framework with regards to language development and name specification. The aim of VHML is to connect some of these different languages. VHML is under development and one objective of this project is to make it XML-based, which is one step further in the process of connecting some of the different languages.

Another objective of the project is to verify, validate and evaluate VHML, which will make the language more solid, homogenous and complete. A significant objective with the development of VHML is to release it to the world. This would be a huge step forward, since it would enable developers to work together in the same directions, using the same markup language.

The objective of developing the DMT is to facilitate the development of the dialogues in interactive TH applications. When using a TH as a user interface within an application you may want it to be able to interact with the user. Having a dialogue management tool would make it easier for the programmers to create correct dialogues. Further, the tool would enable building tree structures of the dialogue. A dialogue management tool is useful when creating any kind of dialogue, for example within an interactive TH application but also in applications using ordinary text based dialogues, such as in applications that maintain *Frequently Asked Questions* (FAQs).

1.3 Problem formulation

In order to reach the aim, the project is divided into three separate, but related, parts;

1. Verify and validate the VHML Working Draft v. 0.1 (VHML v. 0.1, 2001), as well as evaluate the new version of the Working draft, in order to formulate a long-term strategy for the use and development of THs. This was divided into three partial areas:

   - the effect of emotion on speech and speech utterance.
   - the effect of emotion on facial expression and facial gestures.
   - the use of XML as a markup language for controlling VHs.

   VHML involves all languages needed for the implementation of a VH. However, since the project concentrates only on THs, the parts in VHML addressing body animation are excluded.

2. Develop an XML-based Java application, the DMT, for constructing dialogues to be used in interactive TH applications or any other dialogue based application.
3. Demonstrate VHML and the DMT by developing and evaluating two interactive TH applications. (This part was changed during the project and is further discussed in section 5, Talking Head application.)

1.4 Limitations

There are some limitations within which the project was performed. These are:

- *VHML* is the language to be verified for the use of developing THs and the language should be *XML* based.
- The DMT is to be developed using *Java*.
- The underlying structure of DMT is to be a new markup language, the *Dialogue Management Tool Language* (DMTL). DMTL is to be created to suit the dialogue managers that are being developed at Curtin.
- The demonstration applications have to be *interactive*.

1.5 Methodology

This section describes the methodology applied to the three parts mentioned above.

1.5.1 VHML

The first step was to make the language XML-based. In order to do so, a decision was taken to use a DTD, which was created.

The next step was to define a number of criteria for a stable markup language. These criteria constituted a base for the decisions that was taken during the verification and validation of VHML, section 3.1.

The Working Draft v. 0.3 (VHML v. 0.3, 2001) was evaluated in cooperation with the members of the InterFace project.

The outcome of the work is the VHML Working Draft v. 0.4 (VHML v. 0.4, 2001). This document is attached as Appendix A.

1.5.2 DMT

The first step of the development of the DMT was to create the DMTL. This was made in cooperation with the developers of the dialogue managers at Curtin. The reason for this was that the output from the DMT should be a DMTL file and the dialogue managers should be able to use that DMTL file.

The development of the DMT was in Java and documented with JavaDoc v. 1.3. This makes it easier for future programmers who will be working with the maintenance and further development of the DMT. Further, a user manual was created to guide the user when using the tool.

The DMT was tested and an informal evaluation was performed.

Further, a paper concerning the development of the DMT was created for a workshop about THs at the *OZCHI* Conference, held in Fremantle November 20th 2001 (Gustavsson, Strindlund & Wiknertz, 2001). The paper was presented by the project group at the workshop. This document is attached as Appendix B.

1.5.3 Demonstration and evaluation

An initial evaluation of an earlier developed TH application at Curtin, was performed at the TripleS Science Fair, held in Perth August 31st 2001.

A decision was taken to only develop one application, *The Mystery at West Bay Hospital*. This is discussed further in section 5, Talking Head application. An outline of a mystery for the application was written.

To implement the mystery, dialogues for the interaction with the user were created using the DMT. Questions to the application were requested and gained from the members of the Interface group at Curtin.

The mystery application was evaluated and tested by people at Curtin.

2 Literature review

This literature review covers related aspects of interactive *Virtual Human* (VH) and *Talking Head* (TH) technology from the discipline of TH interfaces, facial animation systems, facial gesture, human speech, MPEG-4, XML, VHML and dialogue management.

2.1 Talking Head interfaces

Why is a TH useful as a user interface? One aspect why the THs are useful in computer based presentations is that animated agents that are, for example, based on real video, cartoon-style drawings or model based 3D graphics often make presentations more lively and appealing and therefore make great improvements. They also make the human computer interaction become more like the conversation styles known from human-human communication (André, Rist & Müller, 1998a).

Another important reason for using animated characters is to make the interface more compelling and easier to use. The characters can, for example, be used for attracting the user's focus of attention, to guide the user through several steps in a presentation, to be able to use two-hand pointing or to express nonverbal conversational and emotional signals (André, Rist & Müller, 1998b). Although, it must be noted, they have to perform a reasonable behavior to be useful (Rist, André & Müller, 1997).

Another motivation for using interface agents is that sound, graphics and knowledge can convey ideas faster than technical documents. An individual can often present an idea, feeling or thought in a ten minute long presentation that would otherwise take pages of formal documentation to describe (Bickmore et al., 1998).

Further, when people know what to expect, they can handle their tasks on the computer with greater sense of accomplishment and enjoyment. If a TH is implemented with respect to what people would expect from the same kind of creature in the real world, regarding, for example, politeness, personality and emotion, the better the user interface is (Reeves & Nass, 1996).

What are the drawbacks of using a virtual character as a user interface? A drawback with THs is that the more real the animated character appears, the more expectations the user gets. If the user gets the feeling that he or she is interacting with a human being, the user might get disappointed if the character is not as intelligent as expected. On the other hand, if the TH has a too simple appearance, the user might get bored. The developers of the THs have to balance between these two aspects.

Internet is an area where applications for virtual characters can be successful. The following benefits of using a virtual character has been identified:

- Give a personality to the web page.
- Enable to talk to each person visiting the site, people like to be talked to.
- Make visitors remember main messages better.
- A talking person can be more persuasive than written text (Pandzic, 2001 (to be published)).

When using a TH in an Internet application, several things can be drawbacks if they are not solved nicely. Some people might not feel comfortable in downloading software on

their own computer only to get an unknown improvement of the service, for example, a TH guiding the user through the web pages. The ideal situation is that no installation at all is necessary. Furthermore, most people do not have fast Internet access, so the applications should not require high additional bandwidth. The virtual character also has to be well integrated with all other contents on the web page; text, graphics, forms, buttons etc., to be able to react to the user's actions (Pandzic, 2001 (to be published)). If this is not solved the applications might not be appreciated and thus not be seen as a service improvement.

2.1.1 Applications

There exist several TH applications today. These can be categorized into the following areas; entertainment, personal communications, navigation aid, broadcasting, commerce and education (Pandzic, 2001 (to be published)).

The *Olga* project was a research project aiming to develop an interactive 3D animated talking agent. The goal was to use *Olga* as the user interface in a digital TV set, where *Olga* would guide naive users through new services (Beskow, Elenius & Mc Glashan, 1997). *Olga* was intentionally modeled as a cartoon, with exaggerated proportions as well as some extravagant features, such as antennas, figure 1.

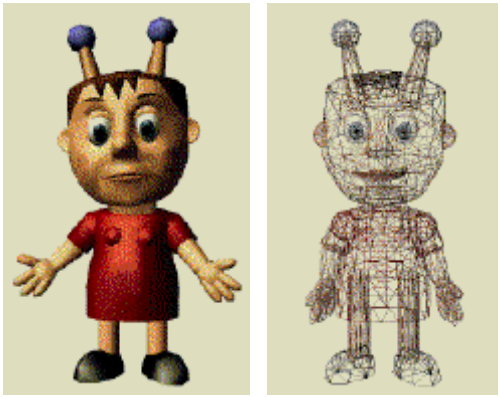


Figure 1. The Olga-character
(Beskow, Elenius & Mc Glashan, 1997). Reproduced by permission.

The main reason for this has to do with what the user expects. If the agent looks exactly as a human being, in a realistic way, the user might get too high expectations of what the system can perform in terms of the system's social, linguistic and intellectual skills. A cartoon on the other hand, does not promote such expectations, since the only experience most people have with cartoons comes from watching them, not *interacting* with them (Beskow, Elenius & Mc Glashan, 1997).

A TH, *August*, has been created for the purpose of acting as an interactive agent in a dialogue system, figure 2. The purpose of the dialogue system is to answer questions within the domains it can handle, for example about Stockholm. To increase the realism and believability of the dialogue system, the TH has been given a great number of communicative gestures such as blinks, nods etc., and also more complex gestures tailored for particular sentences (Lundeberg & Beskow, 1999). Believability is further discussed in section 2.2, Facial animation.

Figure 2. The talking agent August and the 19th century Swedish author August Strindberg (Lundeberg & Beskow, 1999). Reproduced by permission.

Cole et al. (1999) have developed a comprehensive set of tools and technologies, built around an animated TH, *Baldi,* to be used by deaf children in their daily classroom activities. The students interact with *Baldi* through speech, typed input or mouse clicks. *Baldi* responds to their input using auditory visual speech synthesis, i.e. when *Baldi* speaks, the visual speech is presented through facial animation, synchronized with speech that is either synthesized from text or recorded by a human speaker. Using these tools and techniques, teachers and students can design different applications for using *Baldi* in classroom exercises in which students are able to converse and interact with *Baldi*.

The *FAQBot* is a question/answer application that answers a user's questions using knowledge from FAQs. It integrates speech, facial animation and artificial intelligence to be capable of helping a user through a normal question and answer conversation. The *FAQBot* takes users' questions, posed in their own language, and combines an animated human face with synthesized speech to provide an answer from FAQ files. If the agent is being accessed via Internet, it will be able to reply to a user's question with expert knowledge faster than the manual process in finding the answer on Internet would take (Beard, 1999).

Web based virtual characters are being used to deliver jokes and other amusing contents. They are suitable for this because they generally do not require high bandwidth and because they can be implemented to achieve interaction with the user. In that way the user can provoke certain reactions from the character (Pandzic, 2001 (to be published)).

Delivering invitations, birthday wishes, jokes and so on via Internet can be done by sending electronic greeting cards including a talking virtual character (Pandzic, 2001 (to be published)). *LifeFX* is an application that makes it possible to send a VH along with your emails, who speaks the message you have typed. The author of the email is also controlling the emotions being expressed by the VH. You can send facemail with your own voice and in the future you will be able to send a VH created from a picture of yourself (LifeFX, 2001).

The virtual character can be used as a newscaster on the Web. The application might be implemented to remember the user's particular interests and making the virtual character only deliver the news with this content or deliver the news in a certain order depending on these interests. By using this kind of application it is possible to get the news at any time, despite from the TV news that are only being broadcast at certain hours (Pandzic, 2001 (to be published)). *Ananova* is an application of this kind, figure 3.

A TH is presenting news on several different platforms, like mobile devices, PCs, digital TV and interactive kiosks. *Ananova* is providing the option to choose between different news areas. Whenever, for example, a journalist is filing a news story or a goal is scored at a football match the *Ananova* system processes the information and makes it available for being broadcast (Ananova, 2000).

Figure 3. Ananova.
© Ananova Ltd. 2001. Reproduced by permission. All rights reserved.

Further, a virtual character can be used to welcome a visitor to a certain web page as well as guide the user through a number of web pages or to provide hints (Pandzic, 2001 (to be published)).

There exist several applications to be used by companies as the front line customer support on a web page. Currently, most of these applications are text based, possibly displaying an image of a person in order to give it an identity. An animated virtual character is the next logical step for these kinds of applications (Pandzic, 2001 (to be published)).

Only a small number of applications have been described here. Some other existing applications can be found at the Interface web page (Interface, 2001).

THs are a widely growing issue in many different areas. They can be used both as very useful tools and aids, as well as for making an application more amusing. An outcome of this project will be an interactive TH application that belongs to the more amusing category.

One of the goals to achieve while developing a TH is to create a “believable character”, i.e. a character that provides the illusion of life (Bates, 1994). To make a TH believable it is important to be able to animate the character. This is discussed in the following section.

2.2 Facial animation

The most commonly used interface for personification is a human face (Koda & Maes, 1996). The human face is an important and complex communication channel. While talking, a person is rarely still. The face changes expressions constantly (Pelachaud, Badler & Steedman, 1991) and this is something to take into account when developing a TH application.

Initial efforts in representing human facial expressions in computers go back well over 25 years. The earliest work with computer based facial representation was done in the

early 1970's. Parke created the first computer facial animation in 1972 and in 1973 Gilleson developed an interactive system to assemble and edit line drawn facial images. In 1974, Parke proposed a parameterized three-dimensional facial model. In the early 1980's, Platt developed the first physically based muscle controlled face model and Brennan developed techniques for facial caricatures. The short animated film *Tony de Peltrie* appeared in 1985 as a landmark for facial animation, where computer facial expression and speech animation for the first time were a fundamental part of telling a story (IST Programme, 2000).

In the late 1980's, Waters proposed a new muscle based model in which the animation proceeds through the dynamic simulation of deformable facial tissues, with embedded contractile muscles of facial expression rooted in a skull substructure with a hinged jaw. During the same years, an approach to automatic speech synchronization was developed by Lewis and by Hill. The 1990's have seen increasing activity in the development of facial animation techniques. At the UC Santa Cruz Perceptual Science Laboratory, Cohen has developed a visual speech synthesizer; a computer animated talking face incorporating the interaction between nearby speech segments. Recently, the use of computer facial animation as a key story telling component has been illustrated in the films *Toy Story* and *A Bugs Life* produced by Pixar, *AntZ* produced by Lucas Arts (IST Programme, 2000) and *Final Fantasy* produced by Sakaguchi & Sakakibara (2001).

So why should user interfaces with animated humans be preferred to other interfaces? Pandzic, Ostermann & Millen (1999) found in their experiments that users revealed more information, spent more time responding and made fewer mistakes when they were interacting with an animated facial display than with a traditional paper and pencil questionnaire. They also found that a service with facial animation was considered more human like and provoked more positive feelings than a service with only audio. However, if the animated character is to be considered human like it has to be believable. As Bates (1994) said:

> *"If the character does not react emotionally to events, if they don't care, then neither will we. The emotionless character is lifeless, as a machine"*

He also stated that emotion is one of the primary means to achieve believability, because emotions help us to know that the characters truly care about what happens in the world around them. "Believable" is used in the sense of believable characters in the arts. It means that the user can suspend their disbelief and feel that the character is real. It should be pointed out though, this does not mean that the character has to be realistic.

When we interact with other human beings, regardless of our language, cultural background, age etc., we all use our face and hands in the interaction (Cassell, 2000). Blinks and nods are used to communicate nonverbal information such as emotions, attitude, turn taking and to highlight stressed syllables and phrase boundaries (Lundeberg & Beskow, 1999).

Some facial expressions are used to delineate items in a sequence, as punctuation marks do in written text (Pelachaud, Badler & Steedman, 1991). Facial displays can replace sequences of words as well as accompany them. A phrase like "She was dressed" followed by a wrinkled nose and a stuck out tongue would be interpreted as if she was ugly dressed (Ekman, 1979, as referred in Cassell 2000). They can also serve to help disambiguate what is being said when the acoustic signal is degraded (Cassell, 2000),

even though, in optimistic acoustic conditions facial animation does not help understanding (Pandzic, Ostermann & Millen, 1999).

An important issue when we want a character to be capable of communicative and expressive behavior is not just to plan what to communicate but also how to synchronize the verbal and the nonverbal signals (Poggi, Pelachaud & de Rosis, 2000). If the audio and the facial gestures are not synchronized, the character is more likely not to be referred to as believable and human like.

When people speak there is almost always some sort of emotional information included and there are facial expressions that correspond to different emotions. Ekman & Friesen (1975, as referred in Lisetti & Schiano 2000) have proposed six basic emotions that are identified by their corresponding six universal expressions and are referred to with the following linguistic labels; *surprise*, *fear*, *anger*, *disgust*, *sadness* and *happiness*. These emotions are what we refer to as universal emotions. Wierzbicka (1992, as referred in Lisetti & Schiano 2000) though, has found that what we refer to as universal emotions may well be culturally determined. For example, Eskimos have many words for anger, but Ilgnot language of the Philippines or the Ilfaluk language of Micronesia do not have any word corresponding to the English word anger in meaning.

Further, there is a belief that a transition from a happy face to an angry face must pass through a neutral face because these two emotions lie at opposite points in the emotion space. And the same is believed for any two emotions situated in different regions of the emotion space (Lisetti & Schiano, 2000). Therefore, at least a neutral face as well as faces expressing the six different emotions is needed to create a believable facial animated TH.

2.2.1 Reflections

To get a feeling of what facial animation means regarding, for example, a user's engagement, the project group went to see the animated movie, *Final Fantasy* (Sakaguchi & Sakakibara, 2001). The film is totally based on animation; i.e. no real actors are involved in the scenes, although using actor's voices produces speech.

The overall impression of the film was that it was really well created, in some scenes it was even hard to say if it was an animated character or a real human. One good example is *Dr. Sid* in figure 4.

Figure 4. Dr. Sid in Final Fantasy (Sakaguchi & Sakakibara, 2001).

The quality of the different characters varied. Here follows some of the project group's points regarding the quality:

- It seemed as if more details were included in the faces, i.e. beard, wrinkles, noticeable bones and so on, the more real the face appeared.
- The hair was not completely realistic. When the characters were moving, the hair looked somewhat stiff, i.e. it seemed to be moving in separated blocks.
- The filmmakers had managed to catch the reflections of light in the eyes and that made them look very natural.
- The eye contact between the characters was not completely realistic. In some scenes it seemed as if they were not having a natural eye contact when they were talking to each other, as if they looked a little beside the character they were talking to.
- Regarding the body movements, they most of the times looked a little angular and not quite human.
- The skin seemed unnaturally hard. When the characters were touching each other the part that was touched was not affected. It should have moved inwards a little to appear human.
- As explained before, the speech was not automatically produced. Instead, real actor's voices were used. Automatically produced voice is a further step in creating a totally animated film. But more effort could have been made regarding the synchronization between speech and the facial animation, which was a lack sometimes. This is the reaction by several other reviewers as well (Hougland, 2001; Popick, 2001).

Wong (2001) gives hard criticism to the movie. This, according to himself, is probably because the aim of the movie is to be realistic. That makes the viewers, including himself, to expect a lot more of the movie than they would have done if the movie had been an ordinary cartoon. Since the expectations were not met, that could have affected his impression and the criticism he wrote. But even though the animation was not perfect, the fact is that the animation in the movie is very, very good, and several reviewers also point this out, for example by Cardwell (2001). Popick (2001) wrote:

"…the characters are so frighteningly lifelike (especially Dr. Sid) that it becomes distracting…"

A way to animate a TH is to mark up the text to be expressed. In order to do this a predefined language is an extremely useful tool. This is where VHML plays a role by being such a tool. VHML is described in sections 2.7 and 1.

To make the TH as believable as possible it is important to put a great amount of effort in the animation part. The next section describes facial gestures. How changes in the face are achieved in the TH applications used in this project, is described in section 2.4, MPEG-4.

2.3 Facial gestures

Communication is a dynamic process where many components are interacting. When people speak, the sound and the facial expressions are tightly linked together. Thus, for a TH there must exist a program that in advance knows all the rules for how the face should act whilst speaking, in order to generate the motions automatically. Nonverbal cues may provide clarity, meaning or contradiction for a spoken utterance. Therefore, it is impossible to have a realistic or at least a believable, autonomous agent without the influence of all the verbal and nonverbal behaviors (Cassell et al., 1994).

These nonverbal behaviors are not always the same all around the world. For example, shaking one's head can mean to disagree in some parts of the world and to agree in some parts. According to Ekman (1984, as referred in (Pelachaud, Badler & Steedman, 1991) shaking one's head means to agree independently of cultural background. This does not agree with the project group's opinion but in this project no further investigation about this has been made and all examples are taken with respect to our knowledge and interpretation of the behavior of the people in the world.

According to Miller (1981, as referred in Huynh 2000), only 7% of a message are sent through words. The major part of the information is sent through facial expressions, 55%, and vocal intonation, 38%. One reason for this is that humans unconsciously know that nonverbal signals are powerful and primarily express inner feelings that can cause immediate actions or responses. But also because nonverbal messages are more genuine, since the nonverbal behaviors are not as easy to control as spoken words, with exception for some facial expressions and tone of voice. The primary uses of nonverbal behavior in human communication can be put together in five groups:

1. *Expressing emotions.* The message will be more powerful when complementing words with nonverbal behaviors.
2. *Conveying interpersonal attitudes.* Spoken words are easy to control, but nonverbal behaviors will reveal the inner feelings.
3. *Expressing feelings stronger.* For example, if something is too disturbing to express verbally, nonverbal signals can be used instead.
4. *Increasing the possibilities in communications.* Words have limitations that might disappear when gestures and other nonverbal behaviors are used.
5. *Communication cues.* When accompanying speech with nonverbal behavior, turn taking, feedback and attention will follow more easily.

2.3.1 Facial expression

All facial expressions do not necessarily correspond to emotions. In the same way as punctuation does in a written text, some facial movements are used to delineate items in a sequence (Pelachaud, Badler & Steedman, 1991). Ekman (1984, as referred in Pelachaud, Badler & Steedman, 1991) characterized the facial expressions into different areas:

- *Emblems.* Correspond to movements that have a well-known and culturally independent meaning. Can be used instead of common verbal expressions, like nodding instead of saying "I agree".
- *Emotional emblems.* Convey signals about emotions. Are used to refer to an emotion without feeling it, like wrinkle one's nose when talking about disgusting things.
- *Conversational signals.* Punctuate speech in order to emphasize it. Most of the times this involves movements of the eyebrows. For example, raised eyebrows can occur to signal a question.
- *Punctuators.* Correspond to the movements that appear during a pause or to signal punctuation marks, such as commas or exclamation marks. Eye blinks and certain head movements usually occur during pauses. However, the use of

punctuators is emotion dependent, a happy person might, for example, punctuate his speech by smiling.

- *Regulators.* Correspond to how people take turn in a conversation and will help the interaction between the speaker and listener. Duncan (1974) has divided the signals according to what is happening in the conversation;

  Speaker-Turn-Signal is used to hand over the speaking turn to the listener.
  Speaker-State-Signal is displayed at the beginning of a speaking turn.
  Speaker-Within-Turn is emitted when the speaker wants to keep his speaking turn and at the same time assure that the listener is following.
  Speaker-Continuation-Signal will follow the Speaker-Within-Turn.

- *Manipulators.* Correspond to the biological needs of the face, such as blinking to keep the eyes moist.

- *Affect displays.* Express emotions in the face.

To obtain a complete facial animation, all of these movements should be taken under consideration.

2.3.2 Facial parts

When a person talks, it is not only the lips that are moving, but the eyebrows may raise, the eyes may move, the head may turn and so on. The face is divided into three main areas where the facial changes occur (Ekman & Friesen, 1975 as referred in Pelachaud, Badler & Steedman, 1991); the upper part of the face, i.e. the forehead and eyebrows, the eyes and the lower part of the face, i.e. the nose, mouth and chin.

The following parts of a face is affected whilst speaking (Pelachaud, Badler & Steedman, 1994):

- *Eyebrows.* Eyebrow actions are frequently used as conversational signals. They can be used to accentuate a word or to emphasize a sequence of words. They are especially used to indicate questions (Ekman 1979, as referred in Pelachaud, Badler & Steedman, 1996).

- *Eyes.* Eyes are expressing very much information and are always moving in some way. The movements can be defined by the gaze direction, which point they fixate and the duration for this. They are crucial for establishing relationships in a non-verbal way and for communication. Further, the eyes blink frequently, there is normally at least one blink per utterance. There are two types of blinks; the periodic blinks that aim to keep the eyes moist, and the voluntary blinks that emphasize speech, accentuate words or mark a pause (Pelachaud, Badler & Steedman, 1996).

- *Ears.* Humans rarely move their ears, but without ears a face would not look human.

- *Nose.* Nose movements are usually indicating a feeling of disgust, but it is also noticeable that the nostrils are moving during deep respiration and inhalation.

- *Mouth.* The mouth is used to articulate the words and to express emotions. For doing this, the lip motions should be able to open the mouth, stretch the lips, protrude the lips etc.

- *Teeth.* Teeth must be visible to make a face look natural, but they do not move, hence it is only the lips that are moving and then the teeth become more or less visible.
- *Tongue.* The mouth movements often hide the tongue, but the movement of the tongue is essential for verbal communication, for example, to format phonemes such as /l/ and /d/.
- *Cheeks.* The cheeks move when the mouth and the lower parts of the eyes are moving and are therefore changing during many emotional expressions. They also reveal characteristic movements during, for example, whistling.
- *Chin.* The movement of the chin is mainly associated with jaw motions.
- *Head.* Head movements can correspond to emblems, like nodding for agreement and shaking for disagreement, but are also used to maintain the flow of a conversation. Head direction may depend upon affect or may be used to point at something.
- *Hair.* The hair is not moving, but to complete the modeling of a face it is essential to include hair, both on top of the head and the facial hair, such as eyelashes, beard and nose hair.

2.3.3 Synchronism

When linking intonation and facial expressions it is important to synchronize them, which means that changes in speech and the face movements should appear to the user at the same time. To make facial expressions look more natural, the duration of an expression is divided into three parts according to the intensity;

- *Onset duration:* How long the facial display takes to appear.
- *Apex duration:* How long the expression remains in the face.
- *Offset duration:* How long the expression takes to disappear.

The values of these parameters differ for different emotions. For example, the expression of sadness has a long offset and the expression of happiness has a short onset. Figure 5 shows an example of the duration of an expression (Pelachaud, Badler & Steedman, 1996).

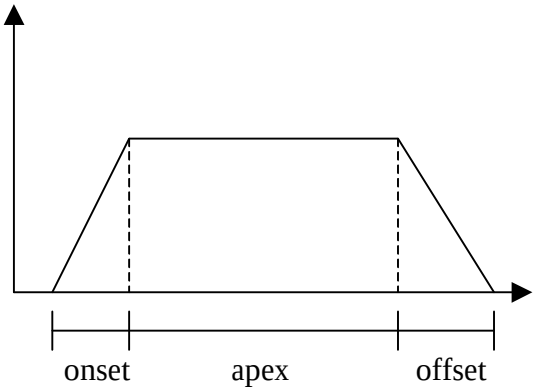


Figure 5. An emotion divided in the three parameters.

Having predefined gestures make it less troublesome for the programmer when creating a human TH. This is one of the features VHML will provide. VHML is

described in sections 2.7 and 1. Facial gestures can for example be implemented by using the standard MPEG-4, which is described in the following section.

2.4 MPEG-4

MPEG-4 is a standard that suits the VHML approach to animate faces, since the expressions can be predefined and relative to each face. Implementing the animation of a TH is not a part of this project. Therefore, this will not be discussed further, but this review is still important since it gives a feeling of how the animation is achieved.

The first step for future facial animation systems was defined in 1998 by the *Moving Picture Experts Group* (MPEG) of the Geneva-based *International Organization of Standardization* (ISO). MPEG-4 provides an international standard that responds to the evolution of technology instead of just specify a standard addressing one application (Shepherdson, 2000). It is an object-based multimedia compression standard, which allows for encoding of different audio and visual objects in the scene independently (Tekalp & Ostermann, 1999).

The representation of synthetic visual objects in MPEG-4 is based on the prior *Virtual Reality Modeling Language* (VRML) standard using nodes, which defines rotation, scale or translation of an object and describes 3D shape of an object by an indexed face set (Tekalp & Ostermann, 1999).

2.4.1 Feature Points

A *Feature Point* (FP) represents a key-point in a human face, like a corner of the mouth or the tip of the nose. MPEG-4 specifies 84 FPs in the neutral face. All of them are used for the calibration of a synthetic face, whilst only some of them are used for the animation of a synthetic face.

The FPs are subdivided into groups according to the region of the face they belong to and are numbered accordingly. Figure 6 shows the FPs on the tongue and the mouth. Only the black points in the figure are used for the animation.

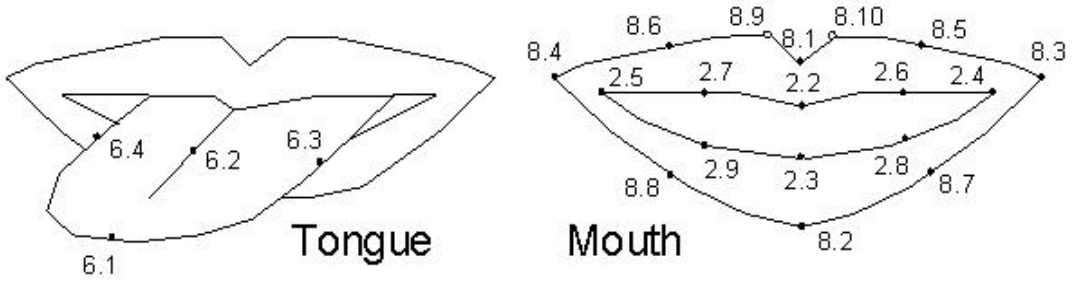


Figure 6. FPs on the tongue and the mouth (ISO/IEC, 1998).

2.4.2 Facial Animation Parameters

The main purpose of the FPs is to provide spatial references for defining *Facial Animation Parameters* (FAPs). FAPs may not affect some FPs, such as the ones along the hairline. However, they are required for defining the shape of a proprietary face model (Tekalp & Ostermann, 1999).

The FAP set includes 68 FAPs; two high-level parameters (FAP 1 and 2) associated with visemes and expressions, and 66 low-level parameters (FAP 3-68) associated with lips, eyes, mouth etc. (ISO/IEC, 1998). The associations are shown in table 1.

Group of FAPs	Number of FAPs
1) visemes and expressions	2
2) jaw, chin, inner lowerlip, cornerlip, midlip	16
3) eyeballs, pupils, eyelids	12
4) eyebrows	8
5) cheeks	4
6) tongue	5
7) head rotation	3
8) outer lip position	10
9) nose	4
10) ears	4

Table 1. FAP groups (Shepherdson, 2000).

High-level FAPs are used to represent the visemes as well as the six most common facial expressions; joy, sadness, anger, fear, disgust and surprise. The emotions and their description are shown in figure 7 and table 2. A viseme is a mouth posture correlated to a phoneme. Only 14 static visemes that are clearly distinguished are included in the standard set. To allow for coarticulation of speech and mouth movement, the shape of the mouth of a speaking human is not only influenced by the current phoneme, but also the previous and the next phoneme (Tekalp & Ostermann, 1999).

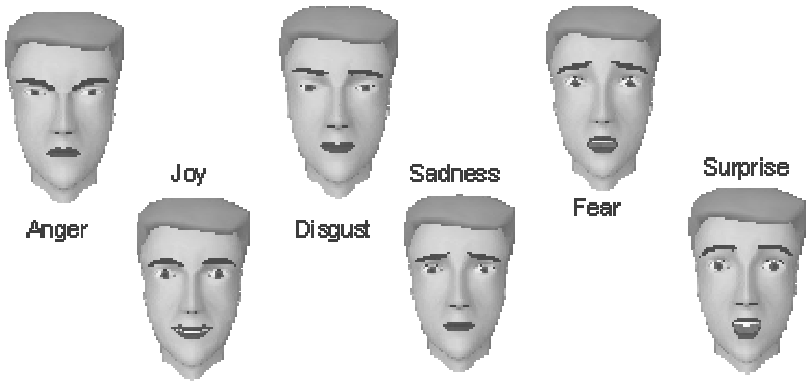


Figure 7. The six different emotions used in MPEG-4 (Tekalp & Ostermann, 1999).

Emotion	Description
Anger	The inner eyebrows are pulled downwards and together, the eyes are wide open and the lips are pressed against each other or opened to expose the teeth.
Joy	The eyebrows are relaxed, the mouth is open and the mouth corners pulled back toward the ears.
Disgust	The eyebrows and eyelids are relaxed and the upper lid is raised and curled, often asymmetrically.
Sadness	The inner eyebrows are bent upward, the eyes are slightly closed and the mouth is relaxed.
Fear	The eyebrows are raised and pulled together, the inner eyebrows are bent upward and the eyes are tense and alert.
Surprise	The eyebrows are raised, the upper eyelids are wide open, the lower relaxed and the jaw is opened.

Table 2. Description of the emotions (Tekalp & Ostermann, 1999).

Low-level FAPs are associated with movements of key facial zones, typically referenced by a FP, as well as with rotation of the head and eyeballs (Pockaj, 1999). Every FAP defines mono-dimensional displacement of the FP with which it is associated (IST Programme, 2000).

Using high-level FAPs together with low-level FAPs that affect the same areas may result in unexpected visual representation of the face. Generally, low-level FAPs have priority over deformations caused by FAP 1 or FAP 2 (Tekalp & Ostermann, 1999).

2.4.3 Neutral face

The neutral face represents the reference posture of a synthetic face. The concept of the neutral face is fundamental. Firstly because all the FAPs describe displacements with respect to the neutral face, but also because the neutral face is used to normalize the FAP values (IST Programme, 2000).

MPEG-4 defines a generic face model in its neutral state by the following properties:

- Gaze is in the direction of the Z axis.
- All face muscles are relaxed.
- Eyelids are tangent to iris.
- The pupils are one third of the diameter of the iris.
- Lips are in contact and the line of the lips is horizontal.
- The mouth is closed and the upper teeth touch the lower ones.
- The tongue is flat and horizontal, with the tip of the tongue touching the boundary between upper and lower teeth (Tekalp & Ostermann, 1999).

2.4.4 Facial Animation Parameter Units

For an MPEG-4 rendering engine to understand the FAP values using its face model, it has to have predefined, model specific, animation rules to produce the facial action corresponding to each FAP. The rendering engine can either use its own animation rules or download a face model and the associated face animation table to get the correct animation behavior. Since the FAPs are required to animate faces of different sizes and proportions, the FAP values are defined in *Facial Animation Parameter Units* (FAPUs). The FAPUs are computed from spatial distances between major facial features on the model in its neutral state, such as, for example, eye separation (Tekalp & Ostermann, 1999).

Six FAPUs have been defined, which are described in table 3 and figure 8 (Tekalp & Ostermann, 1999). The value of the FAP is expressed in terms of fractions of one of the FAPUs. In this way, the amplitude of the movements described by the FAP is automatically adapted to the actual size or shape of the model from which the FAP is animated or extracted (IST Programme, 2000). Rotations are not described by using FAPUs, but are described as fractions of a radian (Pockaj, 1999).

FAPU	Description
AU0	*Angle Unit.* In which angle the face is turned.
ENS0	*Eye – Nose Separation.* The distance from a spot between the eyes down to the tip of the nose.
ES0	*Eye Separation.* The distance between the pupils of the eyes.
IRISD0	*Iris Diameter.* The diameter of iris in a neutral face. By definition, it is equal to the distance between upper and lower eyelid.
MNS0	*Mouth – Nose Separation.* The distance between the tip of the nose down to the mouth.
MW0	*Mouth Width.* The width of the mouth, from one corner to the other.

Table 3. Description of the FAPUs.

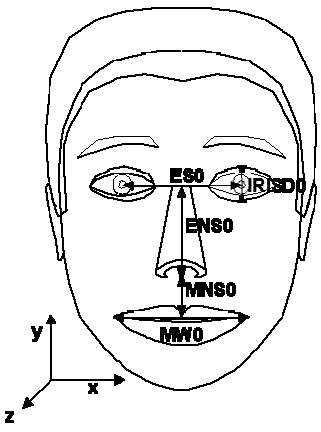


Figure 8. A model showing the FAPUs.

2.4.5 Facial Definition Parameters

The *Facial Definition Parameters* (FDPs) are a very complex set of parameters defined by MPEG-4. They are used for both the calibration of a face and the downloading of a whole face model from the encoder to the decoder (Pockaj, 1999).

A proprietary face model can be built in four steps:

1. Build the shape of the face model and define the location of the FPs on the face model. The model is represented with a mesh of polygons connecting vertices in the 3D space.
2. For each FAP, define how the FPs should move. For most FPs, MPEG-4 only defines the motion in one dimension.
3. Define how the motion of a FP affects its neighboring vertices.
4. For expressions, MPEG-4 provides only qualitative hints on how they should be designed. Visemes are defined as lip shapes that correspond to a certain sound.

When the above steps have been followed, the face model is ready to be animated with MPEG-4 FAPs. Whenever a face model is animated, gender information is provided to the rendering engine. Thus, MPEG-4 does not require using a different face model for male or female gender (Tekalp & Ostermann, 1999).

2.5 Human speech

In a conversation, the vocal expressions do not only tell the listeners the actual meaning of the words, but do also give hints about the emotional state of the speaker, depending on how the words are expressed. The listeners are expecting to hear some vocal effects and are therefore not only paying attention to what is being said, but also in which way it is being said. Children are able to recognize vocal effects even before they can understand any words (Marriott et al., 2000; Stallo, 2000).

When comparing human speech to synthetic speech, the synthetic speech often sounds more machine like, which is a serious drawback for conversational computer systems. Synthetic speech lacks sufficient *intelligibility*, appropriate *prosody* and adequate *expressiveness*. Intelligible phonemes are of importance for word recognition, whilst prosody, i.e. rhythm and intonation, clarifies syntax and semantics as well as gives support to the discourse flow control. Expressiveness, also called affect, gives the listener information about the speaker's mental state and reveals the actual meaning of the words (Cahn, 1990).

The sound of speech depends on the emotions and that has a direct effect on the speech production mechanism. With the arousal of the sympathetic nervous system, for example, with fear, anger or joy, heart rate and blood pressure increase, the mouth can become dry and occasionally there are muscle tremors. Consequently, this will affect how speech is produced (Cahn, 1990).

Further, we deliberately use vocal expression in speech to communicate various meanings. For example, a syllable will stand out because of a sudden pitch change and in consequence of that, the associated word will be highlighted as an important component of that utterance (Dutoit, 1997). If the pitch increases towards the end of a phrase, it denotes that it is a question (Murray, Arnott & Rohwer, 1996, as referred in Stallo 2000). The vocal meaning usually dominates over the verbal meaning. If someone says "Thanks a lot" in an angry tone, it will generally be taken in a negative

way even if the literal meaning of the word is positive. This shows how important the vocal meaning is to avoid misunderstandings (Stallo, 2000).

Since people are very good at recognizing different vocal expressions, acoustic researchers and physiologists have worked to determine speech correlates of emotions. If it is possible to distinguish vocal emotions, there will be acoustic features responsible for it. The problem is that even when a speaking style is consciously adopted, the speech apparatus produces the vocal expressions unconsciously (Scherer, 1996).

Traditionally, three major techniques have been used to investigate speech correlates of emotions (Knapp, 1980; Murray & Arnott, 1993, as referred in Stallo 2000):

1. Actors read neutral, meaningless sentences, letters or numbers and express various emotions.
2. To compare a couple of emotions being studied, the same utterance is expressed in different emotions.
3. The content is totally ignored, either by filtering out the content or by using equipment designed to extract various speech attributes.

The representation of speech correlates of emotion can proceed from either a *speaker model* or an *acoustic model*. In the first approach, the effects of emotion on psychology and on speech are derived from the representation of the speaker's mental state and intentions. The other one describes primarily what the listener hears (Cahn, 1990). The parameters of the acoustic model are grouped into four categories:

- *Pitch.* The intonation of an utterance. Describes the features of the fundamental frequency. The six pitch parameters include pitch average, final lowering, pitch range etc.
- *Timing.* Controls the speed and rhythm of a spoken utterance as well as the duration of emphasized syllables. The five timing parameters include exaggeration, hesitation pauses, speech rate etc.
- *Voice quality*. The overall character of the voice. The seven parameters include breathiness, brilliance, loudness etc.
- *Articulation.* The only parameter is precision, which controls variations in enunciation, from slurred to precise.

The value combinations of these speech parameters are used to express vocal emotion. Table 4 shows a summary of human vocal emotion effects of four of the universal emotions, section 2.2. The parameter descriptions are relative to neutral speech.

	Anger	Happiness	Sadness	Fear
Speech rate	Faster	Slightly faster	Slightly slower	Much faster
Pitch average	Very much higher	Much higher	Slightly lower	Very much lower
Pitch range	Much wider	Much wider	Slightly narrower	Much wider
Intensity	Higher	Higher	Lower	Higher
Pitch changes	Abrupt, downwards, directed contours	Smooth, upward inflections	Downward inflections	Downward terminal inflections
Voice quality	Breathy, chesty tone[1]	Breathy, blaring[1]	Resonant[1]	Irregular voicing[1]
Articulation	Clipped	Slightly slurred	Slurred	Precise

[1]terms used by (Murray & Arnott, 1993)

Table 4. Summary of human vocal emotion effects (Marriott, et al., 2000).

Since the sound of speech supply information besides the actual meanings of the words, it is an important issue to be considered when creating a believable, engaging and interesting VH. Therefore, emotion in speech must be included in VHML. VHML is described in sections 2.7 and 1.

2.6 XML

The *eXtensible Markup Language* (XML) was developed by an XML Working Group formed under the auspices of the *World Wide Web Consortium* (W3C) in 1996 (Bray, 1998). It arose from the recognition that the key components of the original Web infrastructure, such as HTML tagging, simple hypertext linking and hard coded presentation, would not scale up to meet the future needs of the Web (Bosak, 1999). Hopefully, XML will solve some of the Web's biggest problems. For example, the Internet expansion and the fact that it contains a large amount of information, but that it is almost impossible to find what you are looking for when searching the Internet (Bosak & Bray, 1999).

Both these problems arise from the Web's largest language, *HyperText Markup Language* (HTML) (Bosak & Bray, 1999). HTML is easy to learn and is used by many people. Hence, the amount of information published on the Internet grows fast. But HTML does not know what kind of information that is provided, only how it should be presented on a web page. This is what makes it hard to search for the actual information, simply because HTML was not designed for that purpose.

In 1986, the *Standard Generalized Markup Language* (SGML) was approved by ISO as a new markup language (Bosak & Bray, 1999). SGML allows documents to specify what element set to be used within the document and the structural relationships that those elements represent. But SGML is too general, it contains many optional features not needed for web applications (Bosak, 1997).

XML is a "small" version of SGML, to make it easier to define new document types, and to make it easier for programmers to write programs to handle these documents. It omits all the options, and most of the more complex and less used parts of SGML, in return for the benefits of being easier to write applications for, easier to understand and more suited for delivery and interoperability over the web. Nevertheless, it is still SGML, and XML files may still be processed in the same way as any other SGML file (The XML FAQ, 2001).

What are the advantages with XML compared to HTML? First of all, XML is extensible, in the sense that one can define new element and attribute names whenever needed. This cannot be done with HTML. Secondly, XML documents can be nested to any level of complexity, since the author of the document decides the element set and grammar definition. HTML does not support this either. Third, an XML document can be provided with an optimal grammar and use that to validate the structure of the document. This, as well, is not supported by HTML (Bosak, 1997).

What kind of language is XML? As mentioned above, XML stands for eXtensible Markup Language. However, it is not a markup language itself. It is rather a *meta language*, a language for describing other languages. Therefore, XML allows a user to specify the element set and grammar of their own custom markup language that follows the XML specification (Marriott et al., 2000).

2.6.1 The XML document

XML documents, in their simplest form, look very similar to HTML documents. But one difference is that in XML one is able to make one's own rules (Homer, 1999). All XML documents start with an *XML declaration*, like the top row in figure 9. After that declaration, the rest of the document contains *markup*.

```
<?XML version="1.0"?>
<letter type="private">
        <receiver>
                <name>Peter Swan</name>
                <address>
                        <streetaddress>6B Main Street</streetaddress>
                        <city>Sydney</city>
                        <postalcode>7543</postalcode>
                        <state>New South Wales</state>
                        <country>Australia</country>
                </address>
        </receiver>
        <sender>
                <name>Anna Smith</name>
                <address>
                        <streetaddress>76 High Street</streetaddress>
                        <city>Cairns</city>
                        <postalcode>6271</postalcode>
                        <state>Queensland</state>
                        <country>Australia</country>
                </address>
        </sender>
        <message>
                <greeting>Hi Peter</greeting>
                Thank you for …
                <signature>Cheers Anna</signature>
        </message>
</letter>
```

Figure 9. A simple XML document.

Within the markup there are *markup elements* and *character data*. Character data is the actual information in the document, for example, `Peter Swan`, `Sydney` etc., and the markup elements are information about that information, *meta data*, for example **`<name>`**, **`<city>`** etc. The first element, that surrounds all the other elements, is called the *root element* and there can only be one root element within each document. In this example, the root element is **`<letter>`**. Every element that contains some character data must have one *start element*, for example **`<name>`**, and one *end element*, for example **`</name>`**. If the element does not contain any data it is called an *empty element* and could either look like this: **`<name/>`** or like this: **`<name></name>`** (XML White Papers, 2001). XML is case sensitive and hence differs between, for example, **`<name>`** and **`<NAME>`**. The elements can also contain *attribute names* and their corresponding *values*. For example, in the element **`<letter type="private">`** **`type`** is the attribute name and **`private`** the attribute value. The attribute value must be within quotation marks (Homer, 1999).

Character	Entity
&	&
<	<
>	>
”	&quote;
’	'

Table 5. Standard entities in XML.

In order to get the correct XML syntax, XML has reserved some characters. If these characters are to be used within the character data in an XML document, one has to use XML *standard entities* instead. Otherwise, the XML parser does not know what is

character data and what is XML markup, and the XML document becomes unusable. An overview of the standard entities is shown in table 5.

2.6.2 Well formness, validation, DTD and XML Schema

An XML document has to be *well formed*, i.e. its structure has to fulfil specific preconditions to be able to be interpreted and processed correctly in all applications. Some of these criteria are:

- There must be exactly one root element.
- All elements must either have a start element and an end element or be an empty element.
- The order of the elements is hierarchical, i.e. if an element, A, starts within another element, B, then it must also end within that element.
- An attribute must not occur more than once in the same element.
- Attribute values have to be in quotation marks.

An XML document can also be *validated*. In order to get it validated there is a use of a *Document Type Definition* (DTD), in which users can make up their own rules. Rules that describe which elements that are allowed, which attributes they have, of what types the attribute values have to be and in what way the elements can be nested within one another (Bosak & Bray, 1999). An XML document that follows the rules in the DTD is called a valid XML document. However, a DTD is not needed for an XML document to be well formed. But it is useful for authors who want to specify what information a specific type of document should contain.

Another way to build up the grammar for the documents and to validate the documents is to use *XML Schemas*. XML Schema has recently, May 2nd 2001, been approved as a W3C Recommendation (W3C, 1997). DTDs and schemas differ in some ways:

- Schemas are written in XML itself, unlike DTDs that use another syntax.
- DTDs have minimal data constraints available. For example, a **<telephone>** element can be defined to contain **CDATA**, but it cannot, using a DTD, be constrained to just numerals. Schemas allow more specific constraints on data.
- DTD designers are limited to a fixed set of content models. Content models are declarative statements in a DTD that govern what kind of content an element can possess. Schemas provide for archetypes, which allow greater flexibility in limiting and expressing content (Navarro, White & Burman, 2000).

The conclusion from this is that one is able to express more details with the XML Schemas than with a DTD, i.e. new and more specific data types can be constructed. One can also use the archetypes to create some structures that can be reused in many different elements.

The difference in syntax can be both an advantage and a disadvantage. It is an advantage in the way that you can more easily distinguish between writing a DTD and an XML document. But this is also a disadvantage because you have to learn two syntaxes. However, if you are familiar with the two syntaxes this will not be a problem.

One problem with XML Schemas is that it is very new. The project group has not yet found any parsers that can manage the whole syntax of XML and that are free to download. The parsers that have been found are the *Xerces Java Parser*, which is not free,

the *Xerces C++ Parser*, which is neither free nor manages the whole syntax of XML Schema, the *JAXP*, which have no XML Schema support at all (The Apache XML Project, 2001) and the same for *libxml2* (GNOME Mailing Lists, 2001).

2.6.3 XSL Stylesheet

As mentioned, the XML document only contains information. The XML elements do not offer any clues on how this information should be presented on a screen, in a paper or anywhere else. This, in fact, is no disadvantage but rather an advantage for publishers that want to ”write once and publish everywhere”. What XML does, is to make it possible to mark up the content to describe its meaning, without having to worry about how it should be presented to the user. Then it is possible to apply some presentation rules to the document to reformat the content to many different visual mediums. The standard for doing this with XML is to use the *eXtensible Stylesheet Language* (XSL). The latest versions of many web browsers can read the XML document, fetch the suitable stylesheet and use it to sort, format and present the information on the screen (Bosak & Bray, 1999). This can also be used for the processing of VHML elements to various object formats. For example, if the text spoken also should be presented to the user as plain text, XSL can be used to format that text according to the VHML elements used.

2.6.4 DOM and SAX

To process an XML document, an *Application Programming Interface* (API) is used. There are two major types of XML APIs, tree-based and event-based. A tree-based API compiles an XML document into an internal tree structure and then allows an application to navigate that tree. The *Document Object Model* (DOM) working group at the W3C has developed a standard tree-based API for XML and HTML documents. An event-based API reports parsing events, such as the start and end of the elements, directly to the application through callbacks, and does not usually build an internal tree. The application implements handlers to deal with the different events, much like handling events in a graphical user interface (SAX 2.0, 2001). The *Simple API for XML* (SAX), is an event-based API.

SAX requires the least memory and tends to run fast. However, with SAX, the programs see the XML only once and have to figure out what to do with the data straight away, do it and then get ready to handle the next item. DOM, on the other side, is more memory-intensive than SAX, since the entire document must be kept in memory at the same time. The advantage with this is that the programs can go back and fourth in the document and make changes to it (Navarro, White & Burman, 2000). Which one to use depends on what the purpose is. If a fast access is important and there is not much memory available, SAX should be used. If it on the other hand is a need for viewing the whole document more than once, DOM should be used.

2.6.5 XML Namespaces

The flexibility of XML, that makes it possible for users to define their own elements in a document, can also cause conflicts when sharing and blending documents. To prevent these collisions, XML uses *namespaces* (Navarro, White & Burman, 2000). The W3C (1997) defines a namespace by:

An XML Namespace is a collection of names, identified by a URI reference, which are used in XML documents as element types and attribute names.

Before using a namespace, it has to be declared. This is done inside an element with the attribute **xmlns** set to a specific namespace. It can be applied to just a specific element or to the entire document by placing it in the root element of the document. A document can use elements from more than one namespace by blending two or more namespaces. This can be done in two different ways; either by declaring one namespace in the root element and one namespace in another single element, as in figure 10, or by using *qualified names*, as in figure 11 (Navarro, White & Burman, 2000). In both examples the elements **<letter>** and **<reciever>** come from the "foo namespace" and the elements **<sender>** and **<name>** come from the "fee namespace".

```
<?xml version="1.0">
<letter
    xmlns="http://www.foo.com">
   <reciever>
      ...
   </reciever>
   <sender
       xmlns="http://www.fee.com">
      <name>
         ...
      </name>
   </sender>
</letter>
```

Figure 10. Blending namespaces.

```
<?xml version="1.0">
<letter
    xmlns:foo="http://www.foo.com"
    xmlns:fee="http://www.fee.com">
   <foo:reciever>
      ...
   </foo:reciever>
   <fee:sender>
      <name>
         ...
      </name>
   </fee:sender>
</letter>
```

Figure 11. Qualified names.

The idea of qualified names is to provide shortcuts to represent previously declared namespaces. The technique is to declare multiple namespaces in the root element by expanding the attribute with a colon and the name of the namespace. Qualified names are efficient to use when different namespaces are used randomly, otherwise the other alternative is better.

A namespace can be inherited, that is referred to as *scoping*. The scope of a namespace is the element in which it occurs, along with any contained child elements. For example, **<name>** in figure 11 is in the "fee namespace" since that namespace is inherited from the parent **<fee:sender>**. A *default* namespace is the namespace that applies to the element where it is declared and to any child elements contained within that element that do *not* have prefixes to other namespaces of their own. An example of this is shown in figure 12. Here *foo* is the default namespace and hence it does not need to be declared with an own prefix (Navarro, White & Burman, 2000).

```
<?xml version="1.0">
<letter xmlns="http://www.foo.com"
        xmlns:fee="http://www.fee.com">
   <reciever>
      ...
   </reciever>
   <fee:sender>
      <name>
         ...
      </name>
   </fee:sender>
</letter>
```

Figure 12. A default namespace.

Since XML is a growing standard and supports markup languages in a unique way, VHML will be based on XML. As pointed out in the work by Stallo (2000), there are three significant features that additionally emphasize the usefulness of XML when developing VHML; extensibility, structure and validation.

2.7 VHML

The *Virtual Human Markup Language* (VHML) is designed to support the development of VHs in the area of human computer interaction, with regards to facial animation, body animation, dialogue manager interaction, text-to-speech production, emotional representation and hyper and multimedia information (Marriott, Pockaj & Parker, 2001). Although the language is general, the intent is to use it while implementing a TH or a VH interacting with a user via a web page or application. This section is a summary of the VHML Working Draft v. 0.1, written in March 2001 by the Interface group at Curtin (VHML v. 0.1, 2001). It should be pointed out that VHML is not implemented.

This project aims to verify and validate the VHML Working Draft v. 0.1. A number of criteria will be defined and one outcome of the project will be a new version of the VHML Working Draft, where the language as much as possible fulfils these criteria. The new working draft will be evaluated within the project. The work with VHML is described in sections 1, 6.1 and 7.1.1.

The language is based on XML and consists of the following sub languages:

- EML Emotion Markup Language
- SML Speech Markup Language
- FAML Facial Animation Markup Language
- HTML HyperText Markup Language
- BAML Body Animation Markup Language
- DMML Dialogue Manager Markup Language

These sub languages are described later in this section. Given the time constraints for this project, only the head is considered. Therefore, BAML will not be given much effort of improvement, neither will HTML nor DMML.

The rendering system that supports VHML will render an input document that is marked up in VHML as both visual and spoken output. It is responsible for using the information contained in the markup to render the document as intended by the author. The input document may be produced automatically, by human authoring or by a combination of these two. VHML defines the form of that input document.

VHML has the root element **<vhml>**. The other element included on the top level is **<embed>**. Information about the two elements is shown in table 6 and a fragment of a VHML document is shown in figure 13.

Element	Description
vhml	Root element that encapsulates all other vhml elements.
embed	Gives the ability to embed foreign file types such as sound files etc., and for them to be processed properly.

Table 6. Elements in VHML.

```
<vhml>
   This is a simple VHML file with only an embed element.
   <embed type="mml" src="songs/aaf.mml"/>
</vhml>
```

Figure 13. A simple VHML fragment.

In the following sections the sub languages of VHML v. 0.1 are described, i.e. EML, SML, FAML, HTML, BAML and DMML.

2.7.1 EML

The *Emotion Markup Language* (EML) defines the emotion elements that affect the VH regarding voice, face and body, these elements are therefore inherited by the speech and facial animation languages. The elements in EML provide the VH with looks and sounds according to the specified emotion. The elements defined are the following:

- **<anger>**
- **<joy>/<happy>**
- **<neutral>**
- **<sadness>**
- **<fear>**
- **<disgust>**
- **<surprise>**
- **<dazed>**
- **<confused>**
- **<bored>**

There are also other elements in EML, which as well affect the VH regarding voice, face and body. These elements are not emotions but well-known human emotional responses.

- **<agree>**
- **<disagree>**
- **<emphasis>**
- **<smile>**
- **<shrug>**

2.7.2 SML

It is very difficult for a *text to speech* (TTS) synthesizer to make speech sound human with only plain text as input. Since humans are automatically emphasizing important words, pausing for effects and pronouncing foreign words correctly, the speech will sound unnatural and the intelligibility will decrease unless extra information for controlling these parameters are included in the text. The aim of the *Speech Markup Language* (SML) is to define markup elements for controlling this.

The SML in VHML is based on two languages. One of them is the original *Speech Markup Language* (SML) developed by Stallo (2000), which in turn is based on the standard for TTS markup, *Sable* (2001). The other one is the *Speech Synthesis Markup Language* (SSML) , which is a working draft developed by W3C (2001). W3C has estimated that SSML will become a recommendation in early 2002. Therefore, the aim of the new SML is to be as similar to SSML as possible, regarding elements and structure, and that the original SML code should be changed to suit this.

The emotion elements are inherited from EML since they affect speech. The other elements defined in SML are the following:

- **<p>** and **<paragraph>** divide the text into paragraphs.
- **<s>** and **<sentence>** divide the text into sentences.
- **<say_as>** specifies the pronunciation of the contained text by indicating the type of the text.
- **<phoneme>** provides a phonetic pronunciation of the contained text.
- **<voice>** specifies a change in speaking voice.
- **<emphasis>** emphasizes contained text.
- **<break>** controls pausing and other prosodic boundaries between words.
- **<prosody>** controls the pitch, speaking rate and volume of the speech output.
- **<audio>** supports insertion of audio files.
- **<mark>** places a marker into the output stream for a synchronous notification.
- **<emphasise_syllable>** and **<emphasize_syllable>** emphasize a syllable within a word.
- **<pause>** inserts a pause in the utterance.
- **<pitch>** changes pitch properties of contained text.

2.7.3 FAML

To be able to create a TH using facial animation, a *Facial Animation Markup Language* (FAML) has been developed by Huynh (2000). FAML was created for controlling the facial gestures, expressions and emotions in the TH animation for the *FAQBot* application developed by Beard (1999). FAML makes it possible to mark up the input text by specifying type, intensity and duration of the facial gestures, expressions and emotions. The facial display is then synchronized with the speech, to ensure that the animations appear at the right time.

The original FAML is not XML-based. However, the aim of FAML within VHML is that it should be. The emotion elements in FAML are inherited from EML since they affect facial animation. The other elements defined in FAML are described in the following paragraphs.

The look elements turn both the eyes and the head to look in the specified direction.

- **<look_left>**
- **<look_right>**
- **<look_up>**
- **<look_down>**

The head elements only turn the head in the specified direction. The eyes remain looking in their current direction.

- **<head_left>**
- **<head_right>**
- **<head_up>**

- **<head_down>**

The eye elements only turn the eyes to look in the specified direction. The head remains in its current direction.

- **<eye_left>**
- **<eye_right>**
- **<eye_up>**
- **<eye_down>**

The head roll elements roll the head in the specified direction.

- **<head_left_roll>**
- **<head_right_roll>**

The following elements specify the movements of the eyebrows.

- **<eyebrow_up>**
- **<eyebrow_down>**
- **<eyebrow_squeeze>**

The blink elements animate blinks of both eyes.

- **<blink>**
- **<double_blink>**

The wink elements animate winks of the specified eye.

- **<left_wink>**
- **<right_wink>**

2.7.4 HTML

If a VH is not available in an application, HTML can be used for controlling the text instead or it can be used as a complement to the VH. For example, a sentence that is supposed to be spoken in an angry tone might be written with capital letters, bold letters and so on.

It has not yet been decided if VHML should allow the whole set of HTML, XHTML, a subset of HTML or a subset of XHTML.

This sub language will not be given much effort of improvement in this project.

2.7.5 BAML

The *Body Animation Markup Language* (BAML) is a markup language for supporting the body animation of the VH. BAML is the jurisdiction of the body animation partners within the InterFace group.

This sub language will not be given any effort of improvement in this project.

2.7.6 DMML

The *Dialogue Manager Markup Language* (DMML) is a support for creating a question and answer conversation between VHs.

This sub language will not be given any effort of improvement in this project.

However, since dialogue management is the basis of all interaction between users and VHs the next section describes why the dialogues are so important in VH applications and why a tool for creating these dialogues would be useful.

2.8 Dialogue management

In an interactive TH application, there is a need for the TH to be able to converse with the user in some way. For example, a virtual salesperson has to be able to answer the user's questions about certain products. An information provider must answer questions about a certain domain. Furthermore, both have to actively ask questions or at least notify the user when it is unclear what the user really means.

The more intelligent the TH seems to be in the eyes of the user, the more interesting it will be to interact with. There are several tricks for making an agent seem more intelligent. The chatterbot *Eliza* tricks the user to direct the course of the conversation. In that way *Eliza* does not have to contribute with much substance in the dialogue (Weizenbaum, 1976, as referred in Marriott, Pockaj & Parker, 2001). For example:

> *User*: My mother is always working.
> *Eliza*: Who else in your family is always working?

Eliza's response seems to be intelligent and caring to the user, although it is only conveyed by ordinary pattern matching.

Developing a dialogue includes creating *stimuli* and *responses*. When the user input matches a stimulus, this should trigger the correct response. Depending on the stimulus, the dialogue should traverse into different *states*. This is another well-known trick to make an application seem more intelligent. By handling this, the application will know the context of the dialogue and will therefore be able to respond correctly. The trick has been used by, for example, *Julia* and *Colin*, who are two chatterbots developed by Mauldin (1994). They seem somewhat intelligent to the user even though the structure of their knowledge is an ordinary network with a number of states.

The TH in the following dialogue between a TH and Anna uses the same trick:

> *TH says*, "How are you?" *to Anna*.
> *Anna says*, "Not so good." *to TH*.
> *TH says*, "Why is that?" *to Anna*.
> *Anna says*, "I have a terrible headache." *to TH*.
> *TH says*, "Have you taken aspirin?" *to Anna.*
> *Anna says*, "I have to go. Goodbye!" *to TH*.

Figure 14 represents a fragment of the rules used by the TH in the discussion. S represents the stimulus, written in a regular expression, and R represents the response.

The first question is an active prompt from the TH and does not have to be triggered by a stimulus. Anna's answer, "Not so good." is a stimulus that moves the dialogue to a different state. In this new state, the TH "knows" that Anna is not feeling good. The TH then asks: "Why is that?", which is a response that only can take place because of the fact that the TH "remembers" the previous questions and answers. Anna's answer

about the headache is yet another stimulus that moves the dialogue into a new state and a responding question is posed. Anna's end phrase moves the dialogue into a final state, which also is an entry state and therefore can be entered at any time during the dialogue.

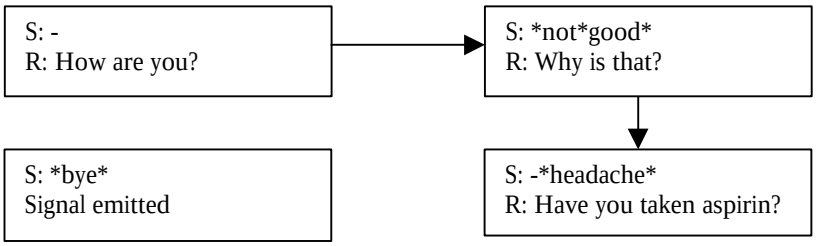


Figure 14. A diagram of the greeting example.

This short example points out the importance of dividing the dialogue into different states. The question "Why is that?" can not be posed without a known context, since it would not have a meaning if the context is missing. Furthermore, to pose the question "Have you taken aspirin?" the TH has to know that Anna suffers from a headache. It is also important to point out that the TH can keep track of a whole sequence of stimuli and responses. This means that the TH can produce a response that relates to a discussion that appeared earlier in the conversation.

The user input might be grammatically incorrect, but it should still match a stimulus that triggers a response. Using pattern matching for the input solves this. Furthermore, a certain response might be considered the "correct" one for more than one input. In the previous example, the input "Not so good." should trigger the same response as for example "I'm not feeling very well today." and hence give the same answer, "Why is that?". By forming regular expressions or word graphs for the *Dialogue Manager* (DM) to parse, it is possible to create a stimulus that matches a great number of user inputs. For example, the stimulus "*not*good*" matches both "Not so good" and "I'm not feeling that good".

Managing the dialogue is a very important issue in order to create an interesting and interactive TH application. By using network structures for the dialogue, it is possible to create a more intelligent conversation since it gives the possibility to keep track of the conversation's state. Since the dialogues might become very large and complex, it can take a great amount of time to construct correct network structures. The aim of this project includes creating a tool that simplifies the construction and maintenance of this kind of dialogues.

3 Virtual Human Markup Language

The *Virtual Human Markup Language* (VHML) Working Draft v. 0.1 of 13th March 2001 (VHML v. 0.1, 2001), created by the Interface group at Curtin and summarized in section 2.7, has been verified and validated. This process is described in the following sections and has lead to a new working draft, version 0.3 (VHML v. 0.3, 2001). The working draft was evaluated, section 6.1, which resulted in version 0.4 (VHML v. 0.4, 2001). The final working draft can be found in Appendix A.

3.1 Criteria for a stable markup language

When designing a new markup language there are several criteria to be considered. During the verification and validation of VHML, seven criteria were defined and used as the basis for all the decisions taken when improving the language. These criteria are:

- *Completeness.* The language must be complete or constructed in a way that is easy to expand.
- *Simplicity.* The language should aim to be as simple as possible and exclude any ambiguous features. That would keep the language fairly small and comprehensive. Nevertheless, this should not affect the previous criterion. In order to fulfil this criterion, elements that have the same functionality should be merged.
- *Consistency.* The language must be consistent in order to make it easier for the user to learn, i.e. the syntax should follow a certain pattern. For example, the element names should be in the same form and have the same kind of attributes.
- *Intuitivity.* The language should aim to be intuitive, thus the user will not always need to consult the specification to be able to use the language. The names of the elements and attributes should be self-describing.
- *Abstraction.* The language should use a high abstraction level. That will make the language easier to understand and thus to use.
- *Usability*. The language should aim to provide features that suit both beginners and advanced users.
- *Standardization.* The language should aim to follow existing standards for the different parts of VHML. It is important that the languages it follows are, or will become, a standard. In case it is probable that it will become a standard, it is important to provide features so the language easily can be changed to follow the standard in the future.

3.2 General issues

One of the aims of VHML was to make it XML-based. That means that a VHML document should be a well formed XML document. In order to not only write well formed but also valid documents, a way to construct the grammar for the documents was needed. There are two ways of writing grammars; by using either a DTD or an XML Schema, as discussed in section 2.6.2. Both ways have advantages and disadvantages. Schemas give a more powerful and richer way of describing information,

but at the time when the decision whether to use a DTD or a schema had to be taken, the project group had not found any parser for schemas that was free to download and could manage the whole syntax of XML. The cost was an important issue for Curtin and therefore a decision was taken to use a DTD, even though that limited the design possibilities. Yet another reason to choose a DTD was that the speech part of VHML is based upon SSML and SSML uses a DTD to validate its documents. Therefore, using a DTD for validating VHML documents will facilitate inheriting new elements from SSML by using XML Namespaces, section 2.6.5. The advantage of this is that if SSML changes, these changes will affect VHML as well. Though, at present SSML is only a working draft, which means that the SSML elements do not exist in the way that they can be inherited by using XML Namespaces. Therefore, this has not been considered for this version of VHML. The VHML DTD is included as Appendix C.

This is an example of a complete VHML document, where a male TH in a happy way describes the weather. He is also looking towards the sky while he is emphasising that there are no clouds at all. The TH is nodding his head when he is making the conclusion that the weather is perfect for a day at the beach.

```
<?xml version="1.0">
<!DOCTYPE vhml SYSTEM "http://www.vhml.org/vhml.dtd">
<vhml>
   <person age="30" gender="male">
      <paragraph>
         <happy>
            I think that this is a great day.
            <smile duration="2s" wait="1s"/>
            <look-up>
               Look at the sky. There is
               <emphasis level="strong">
                  not a single
               </emphasis>
               cloud.
            </look-up>
            <agree duration="3500ms" repeat="4"/>
            The weather is perfect for a day at the beach.
         </happy>
      </paragraph>
   </person>
</vhml>
```

Since VHML aims to be used worldwide and not only in English speaking countries, an additional feature of the language has been considered; to be able to write the elements in any language or using synonyms for the words. For example, it should be possible to use the Swedish word, **<arg>** instead of the English word **<angry>**, which is the name of the element in VHML, and a synonym, **<joyful>** instead of **<happy>**. A solution to this is to use the transform classes inside the *javax.xml.transform.dom* library (XML Standard API, 2001). An overview of how this will work with a Swedish markup is presented in figure 15. The input is a DOM tree of the document and by using an XSL Stylesheet the original DOM tree transforms to a new one that contains the correct element and attribute names, which can then be validated by the DTD. A specific stylesheet has to be constructed for each language as well as for synonyms. For further details about XSL Stylesheets and DOM trees see sections 2.6.3 and 2.6.4.

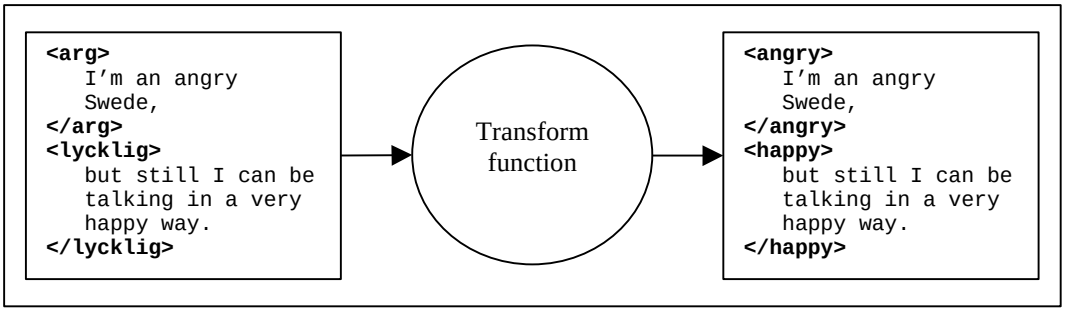


Figure 15. An example on how the transform function works from Swedish to English.

VHML is now composed of seven instead of six sub languages. These will be described in separate sections. The sub languages are:

- Emotion Markup Language (EML)
- Gesture Markup Language (GML)
- Speech Markup Language (SML)
- Facial Animation Markup Language (FAML)
- Body Animation Markup Language (BAML)
- eXtensible HyperText Markup Language (XHTML)
- Dialogue Manager Markup Language (DMML)

VHML can be partitioned in three levels. Figure 16 shows this as well as which sub languages that belong to a certain level.

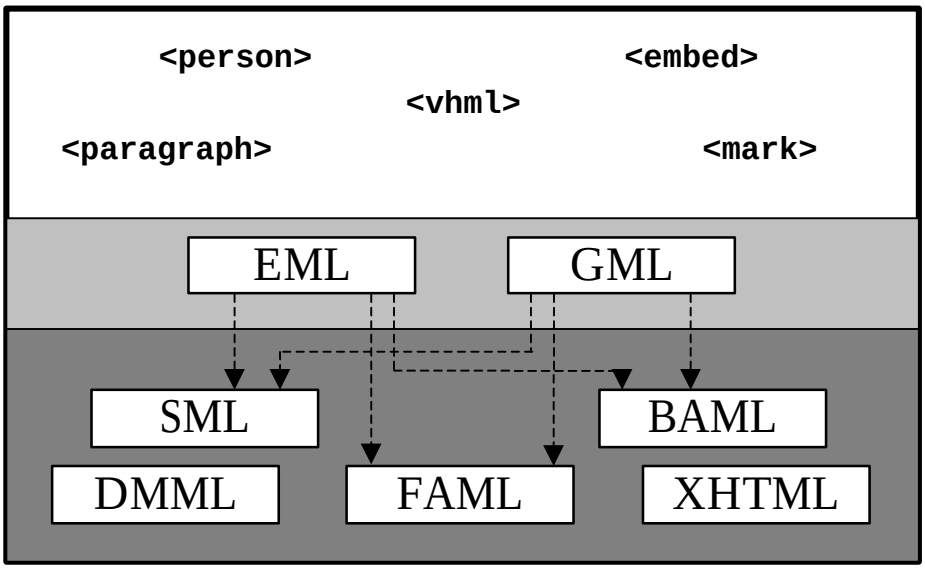


Figure 16. The structure of VHML.

Five elements are not a part of any sub language and belong to the top level of VHML. EML and GML constitute a middle level, since their elements are inherited by some of the other sub languages. The dotted arrows imply inheritance between sub languages. The five remaining sub languages are parts of the lowest level of VHML.

One of the sub languages, SML, is directly based on SSML (W3C, 2001). The reason for this is that SSML is likely to become a standard for speech markup languages and hence it is profitable to keep SML as similar as possible to SSML. This also adapts

VHML to the criterion standardization. Since some of the features in SSML not only affect speech, the changes have touched other elements as well.

Initially, there was not only SSML but also a number of other languages involved when taking all decisions about SML. A detailed comparison was made between:

- the first version of VHML, which included SML made by Stallo (2000) that is already implemented at Curtin.
- Sable (2001) that is an existing standard for text-to-speech markup and constituted a base for SML made by Stallo (2000).
- VoiceXML (2000) that is a speech markup language made by W3C.
- SSML that is already mentioned, which originally is based on Sable and VoiceXML.

Throughout the project, the project group was more and more requested by the Interface group at Curtin to follow the working draft of the SSML specification and therefore all decisions were finally taken based on SSML, even if the solution not always was found to be the best one. Though, VoiceXML is a standard for speech markup, which has a pointer to SSML and hence will be changed according to changes in SSML. This shows that the decision taken to follow SSML was appropriate. Only when SSML did not give any solution, the other languages were considered.

3.3 The top level elements

The elements that can be used at the top level are summarized in table 7 and how the elements are nested is described below.

Element	Description
vhml	Root element that encapsulates all other VHML elements.
person	Specifies the speaker of the document.
paragraph = p	Divides text into paragraphs.
embed	Embeds foreign file types in a VHML document.
mark	Places a marker into the output stream for a synchronous notification.

Table 7. A summary and description of the top level elements.

VHML uses **<vhml>** as root element, which encapsulates all other VHML elements. The root element can contain zero or more **<person>** elements and if there is no **<person>**, one ore more **<paragraph>** elements. Each **<person>** element must contain at least one **<paragraph>** element, which in turn contains elements on a lower level. To imply a paragraph, either **<p>** or **<paragraph>** can be used. This is a feature that follows SSML and that provides a shortcut for typing an element that is used often. However, since a VHML document is an XML document one cannot blend **<p>** and **<paragraph>**. The start and end elements have to be the same.

When humans talk, a specific prosody is used in a sentence that forms a melody in speech and this is the reason why it does not sound robotic when talking. In SSML a **<sentence>** element is used in order to divide the text into sentences to make the speech sound natural. However, the system behind VHML is responsible for inferring the structure by automated analysis of the text, often using punctuation and other language specific data (VHML v. 0.1, 2001). In this way, the text does not have to be divided into smaller parts than paragraphs and the **<sentence>** element becomes useless and was removed from the language. Additionally, the VHML document will remain clearer without having to mark up all sentences with elements and hence make

the language more user friendly as well as increase the simplicity of the language. Since SSML uses a **<sentence>** element there will be a problem when validating an SSML document according to the VHML DTD. Therefore, if SSML becomes a standard and continues using **<sentence>**, it must be restored to VHML.

Several elements in SSML have an attribute **xml:lang** to indicate the language of the enclosing text. VHML allows this attribute only for the **<vhml>** and **<paragraph>** elements. Since the language of the document should not change very often, it should be specified on a higher level.

The **<person>** element is a way of specifying the general speaker of the document regarding gender, age and category. Different variants of a speaker with the same properties can be used and it is also possible to give a defined speaker a name that can be used later in the document. Further, the user has the opportunity to choose a disposition for the speaker in order to decide if the voice generally should be happy, angry, sad etc. In the future; it can be of interest to add even more properties, like physique and nationality/culture, since these, among many other properties, can affect how the VH acts in terms of the face, body and voice. For example, some nationalities or cultures shake their head instead of nod in order to agree, section 2.3, Facial gestures. However, this will not be a part of the present version of VHML. Though, since the **<person>** element is included, the language caters for the change.

<person> should affect the voice as well as the facial animation and in the future also other parts of the body. A child not only sounds different but also acts in another way than an adult, for example when being angry or shaking their head for disagreement. The element can only occur outside the **<paragraph>** elements. If a change in the voice is wanted for only a certain phrase, the **<voice>** element at a lower level should be used.

The **<embed>** element gives the ability to embed foreign file types within a VHML document. At present, there are only two sorts of files that can be embedded; audio and *Music Markup Language* (MML) files (MML, 1999). Though, many other types could be of interest, for example MP3, JPEG, GIF etc. To decide which types of files that should be possible to embed is up to the programmer implementing VHML and will therefore be considered future work.

There exist two ways of setting an arbitrary mark at a given place in a text. All elements, except those on the top level, have an attribute **mark** that can be used. If a mark has to be set between two tags or at the top level, the element **<mark>** could be used. To have two alternative ways of doing something can be seen as decreasing the consistency of the language, but being able to use **mark** as an attribute and not only an element increases the simplicity of the language. The documents will be shorter and hence become more readable when marking something using an attribute instead of including a new element. Since the element not affects the sound, the **<mark>** element has been moved from being an SML element in the former version to instead be a part of the top level.

Figure 17 shows an example on how the high level elements can be used.

```
<?xml version="1.0"?>
<!DOCTYPE vhml SYSTEM "http://www.vhml.org/DTD/vhml.dtd">
<vhml xml:lang="en-US">
   <person gender="female" disposition="happy">
      <p>I'm a woman.</p>
      <p>I've had a great day! Listen to this song.</p>
      <embed type="mml" src="songs/Halleluja.mml"/>
   </person>
   <person category="child" gender="male">
      <mark name="now"/>
      <p>Now I instead talk with the same voice as my son.</p>
   </person>
</vhml>
```

Figure 17. An example of a VHML document, only using the top level elements.

3.4 Emotion Markup Language

The *Emotion Markup Language* (EML) is used for adding emotions into the VH. The language affect the face as well as the body and speech.

There exist hundreds of emotions from which to choose, some are very similar and hard to distinguish, some are seldom used and some are just non expressible feelings and impossible to produce in a VH. The selection of emotions to include in EML is based on what has been done previously in this area, as well as on the universal emotions, which are shown from researches to be clearly and unambiguously expressible, section 2.2, Facial Animation. However, EML is a sub language that can easily be expanded. Different emotions are of importance depending on in which domain the language will be used.

Element	Description
afraid	The eyebrows are raised and pulled together, the inner eyebrows are bent upward and the eyes are tense and alert.
angry	The inner eyebrows are pulled downward and together, the eyes are wide open and the lips are pressed against each other or opened to expose the teeth. The speech rate and the pitch of stressed vowels are increased and the average pitch and pitch range are decreased.
confused	The eyebrows are bent upwards, the inner eyebrows are moving greatly and the corners of the mouth are close together.
dazed	The eyebrows are slightly raised, the eyes opened somewhat wider than normal and the lips are slightly pulled down and outwards.
disgusted	The eyebrows and eyelids are relaxed and the upper lip is raised and curled, often asymmetrically.
happy	The eyebrows are relaxed, the mouth is open and the mouth corners pulled back towards the ears. The speech rate, average pitch and pitch range are increased, so is the duration of the stressed vowels. The changes in pitch between phonemes are eliminated and the amount of pitch fall at the end of an utterance is reduced.
neutral	All face muscles are relaxed, the eyelids are tangent to iris, lips are in contact, the mouth is closed and the line of the lips is horizontal.
sad	The inner eyebrows are bent upward, the eyes slightly closed and the mouth relaxed. The speech rate, average pitch and pitch range are decreased. Abrupt changes in pitch between phonemes are eliminated and pauses are added after long words. The pitch for words before a pause is lowered and all utterances are lowered at the end.
surprised	The eyebrows are raised, the upper eyelids are wide open, the lower are relaxed and the jaw is open.
default-emotion	The emotion specified in the person element or by the application.

Table 8. A summary and description of the emotion elements.

Some of the emotions are currently only defined for the facial animation, though these will also affect the body and speech. Extensive research has to be made in order to find out how the body and speech change under a certain emotion, before the emotion can be added to EML. The emotions that currently can be expressed by a VH using EML are summarized in table 8. How the voice changes is only described for the elements that are already implemented for speech. The body movements are not implemented at all and therefore not described.

To keep consistency in the language, a decision about how the emotion elements should be named had to be taken. Should the elements be expressed as nouns like happiness, anger and sadness, or as adjectives like happy, angry and sad? Some of the already existing markup languages that direct emotions have been investigated.

- *Sony Computer Entertainment Europe* (SCEE) has used a markup language in the *Getaway* project, which is using nouns to describe emotions (Moore, 2001).
- The *Human Markup Language* (HumanML) is a proposed OASIS XML specification and is using nouns for the emotions (HumanMarkup.org, 2001).
- The *Multimodal Presentation Markup Language* (MPML) uses adjectives for the emotion elements, but some of the elements, like angry and surprised, also allows the corresponding noun element, anger and surprise (Ishizuka, 2001).
- The *Facial Animation Coding System Markup Language* (FACSML) uses nouns for the emotions (Binsted, 1998).

The conclusion from this is that there is no existing standard in which form the emotion elements shall be named, though it is more common to use the noun form instead of the adjective form. Additionally MPEG-4, that is the standard often used when animating the face of a VH and what is used within this project, is using nouns for the emotions.

Another important part when designing VHML has been to make it as intuitive as possible, which means that only by looking at the element it should be obvious what the element does. In that aspect, the project group felt that is was most natural to use adjectives as element names, since the face should *look happy* etc. instead of thinking in terms of *happiness should be expressed* in the face. Another advantage of using adjectives is that when reading the VHML document the text will float better if using how the person feels instead of what it expresses. For example, "… when I woke up I realised that **<happy>** today is my birthday **</happy>**…" floats better than "… when I woke up I realised that **<happiness>** today is my birthday **</happiness>**…".

Before taking a decision, an email with a question about what to use was sent off to InterFace as well as to the Interface group at Curtin. The respond was not very good, though some opinions arose. All of those expressed that adjectives sound better, but if the emotion should be used as a value for an attribute, as it is for **<person>** with the attribute **disposition**, then noun would be the best alternative. If the emotion will have any attribute, like **duration** and **intensity**, then it also would sound better using nouns instead of adjectives.

The noun form and the adjective form were compared for each emotion, in order to find the most suitable words. The words that were found are summarized in table 9.

Noun	Adjective
anger	angry
confusion	confused
-	dazed
disgust	disgusted
fear	afraid
happiness	happy
-	neutral
sadness	sad
surprise	surprised

Table 9. A comparison between nouns and adjectives for the emotion names.

Difficulties arose when trying to find the noun word for neutral and dazed. Fear is one of the universal emotions, but afraid was considered a better word to use than fearful, which is the adjective for fear.

All information was summarized and a decision was taken to use adjectives for the emotion elements. Though, some confusion can occur when people are using MPEG-4 and VHML at the same time, since the emotions are in different forms. This problem can be solved by using the transform function that was discussed in section 3.2, General issues, or by simply allowing both by having two copies of each element, one for adjective and one for noun, in the DTD.

The **<default-emotion>** element is a new element that has been added to this version of EML. When the **disposition** attribute of a **<person>** element has been provided, this emotion will be connected to **<default-emotion>** in the rest of the document. If there is no **disposition** specified, the emotion specified by the application will be connected to the **<default-emotion>**. The **<default-emotion>** can be used for returning to the general emotion in the document. However, this can also be done by not specifying any emotion at all for the text.

A new feature was added to the language after a couple of comments on VHML received from Ania Wojdel, a Polish researcher working with facial animation. This was to add a **wait** attribute to all EML, GML and FAML elements in order to make a pause after starting an action and before continuing with further elements or plain text. This could for example, be used when the VH should look angry for a period of time before it starts to talk or when a sigh should start some seconds or milliseconds before a shake for disagreement.

Figure 18 shows how the emotion elements can be used in a VHML document.

```
<?xml version="1.0"?>
<!DOCTYPE vhml SYSTEM "http://www.vhml.org/DTD/vhml.dtd">
<vhml>
   <person disposition=”angry”>
      <p>
         First I speak with an angry voice and look very angry,
         <surprised intensity=”50”>
            but suddenly I change to look more surprised.
         </surprised>
         <happy wait=”2s”>
               Then I change to become very happy instead. The happiness was
               expressed in two seconds before I started to talk.
         </happy>
         <default-emotion>
            The happiness doesn’t last for long and now I’m angry again.
         </default-emotion>
      </p>
   </person>
</vhml>
```

Figure 18. An example of a VHML document using emotion elements.

3.5 Gesture Markup Language

A new language, the *Gesture Markup Language* (GML), was defined in order to include all the elements that in some way control gestures. Previously, these elements were a part of EML and were called emotional responses. However, since not all of them are responses and not only depend on emotions, the elements were separated from EML to build a new sub language of VHML, the GML.

Element	Description
agree	Animates a nod. It is broken into two sections, the head raises and then the head lowers.
concentrate	The eyebrows are lowered and the eyes partly closed.
disagree	Animates a shake of the head.
emphasis	Similar to the agree element, but even the eyebrows are lowered. The text to be spoken is stressed. The pitch and duration values are changed.
shrug	The head is tilted back, the corners of the mouth pulled downward and the inner eyebrows are tilted upwards and squeezed together.
sigh	The cheeks are puffed and also the eyebrows, head and mouth are affected.
smile	Animates the expression of a smile, the mouth is widened and the corners pulled back towards the ears.

Table 10. A summary and description of the GML elements.

In version 0.3 of VHML, the gesture elements only affect the visual animation, although not **<emphasis>**, which also affects speech. Therefore, only the facial movements, except **<emphasis>**, are described in table 10. Some of the other elements could affect speech as well, for example, when a person disagrees the prosody might change in a certain way. Further, some of the elements could affect the whole body, for example, a shrug might raise the shoulders. This should be taken under consideration in future development of VHML.

GML is only a small subset of all gestures that a person might perform. These were selected because of previous work in SML (Stallo, 2000) and FAML (Huynh, 2000), where only these were defined and implemented. Additional gestures that in the future should be considered for being a part of GML are yawn, whistle, think, laugh, cry etc. For example, **<think>** would be a very useful element where a speaker looks thoughtful, while a voice is speaking.

Not many changes have been made for the elements in this sub language from the last version of VHML. Though, a new attribute **repeat** has been added for some of the elements, **<agree>**, **<disagree>**, **<sigh>** and **<shrug>**, in order to make it possible to repeat the action without having to include the element more than once. This is a way to keep the language simple.

Figure 19 shows an example on how the gesture elements can be used in a VHML document.

```
<?xml version="1.0"?>
<!DOCTYPE vhml SYSTEM "http://www.vhml.org/DTD/vhml.dtd">
<vhml>
   <p>
      <emphasis>
           How many times do I have to tell you to make your bed!
      </emphasis>
      <sigh duration="1500ms" wait="1s"/>Stop picking on me, but
      <agree intensity="low">you are right, I will make my bed now.</agree>
   </p>
</vhml>
```

Figure 19. An example of a VHML document using gesture elements.

3.6 Facial Animation Markup Language

The *Facial Animation Markup Language* (FAML) is only used for animating the face, hence the body and voice will not be affected. The elements that can be used are described in table 11. All the emotions and gestures also affect the facial animation and those elements are therefore inherited from EML and GML.

In order to follow the same syntax as SML and SSML, the underline of the element names, in VHML Working Draft v. 0.1, has become a hyphen. This makes the language more consistent and standardized.

The element in version 0.1 called **<blink>** was expanded to **<eye-blink>**, so that all elements regarding the eyes would be grouped together when sorting the elements in alphabetic order in a specification, which will make it easier for the user. Further, all elements should be named in the same way, i.e. elements affecting the eyes should start with the word eye. The user should be able to guess the right name without having to consult the specification. This applies to the intuitive criterion for VHML.

Element	Description
look-left	Turns both the eyes and the head to the left.
look-right	Turns both the eyes and the head to the right.
look-up	Turns both the eyes and the head upwards.
look-down	Turns both the eyes and the head downwards.
eyes-left	Only the eyes turn left, the head remains in its current position.
eyes-right	Only the eyes turn right, the head remains in its current position.
eyes-up	Only the eyes turn upwards, the head remains in its current position.
eyes-down	Only the eyes turn downwards, the head remains in its current position.
head-left	Only the head turns left, the eyes remain in their current positions.
head-right	Only the head turns right, the eyes remain in their current positions.
head-up	Only the head turns upwards, the eyes remain in their current positions.
head-down	Only the head turns downwards, the eyes remain in their current positions.
head-roll-left	A roll to the left in the axial plane.
head-roll-right	A roll to the right in the axial plane.
eyebrow-up	Vertical eyebrow movement upward.
eyebrow-down	Vertical eyebrow movement downward.
eye-blink	A blink of both eyes, which affects both the upper and lower lid.
wink	A blink with one eye as well as movement of the head, outer eyebrow and cheek.
jaw-open	Opens up the jaw.
jaw-close	Closes the jaw.

Table 11. A summary and description of the FAML elements.

Although the eyes and head only can move in four directions; left, right, upwards and downwards, they will have a full range of orientation. The solution is that the elements can be combined. For example, to look at the top left, a combination of **<look-left>** and **<look-up>** can be used. The attribute **intensity** can at the same time be used to make the VH look up and only slightly to the left.

A discussion was held to find a way to increase the simplicity of the language by merging the **<look-XXX>**, **<eyes-XXX>** and **<head-XXX>** elements together in some way. Since it is not possible to make a movement to the right at the same time as to the left, the **<XXX-left>** and **<XXX-right>** elements could be merged to one element **<XXX-horizontal>** using a new attribute, **direction**, that specifies in which horizontal direction the movement should be done. The same would be possible for the vertical movements, which also would lead to that **<XXX-up>** and **<XXX-down>** would be merged to a **<XXX-vertical>** element with the attribute **direction**. One

option was to go one step even further and merge all four elements into one element for each movement, i.e. **<eyes>**, **<head>** and **<look>**. Within this option the three remaining elements must have two new attributes, one that specifies the horizontal direction and one that specifies the vertical direction. The reason for taking the decision to use separate elements for each direction was that it turned out to increase the intuitivity as well as the simplicity of the language, since writing the additional attributes would be even more demanding than writing two different elements.

Using the **<look-XXX>** elements instead of specifying the head and eyes separately is a way of abstracting the language and hence increase the readability of the document. It would be hard to understand what is happening if looking at the bottom right had to be defined by four elements, i.e. **<head-down><eyes-down><head-right><eyes-right>** instead of only using two, i.e. **<look-down><look-right>**. It is also more convenient for the programmer and will not involve any additional problems since the eyes and head are moving at the same rate when looking at something. However, the user can choose to do either way because their meanings are exactly the same. This caters for the usability of the language.

The eyes are not able to move independently of each other, since no situation was found when this could be useful. Instead the VH would only look strange if the eyes moved in different directions. However, the language should be flexible and this will set limitations of the eye movements, as for cross-eyed effects. Thus, the language is designed so that it in the future will be easy to add another attribute, **which**, to the eye elements in order to specify if it is the right, left or both eyes that should move.

It is worth noticing when implementing the head movements that the angle within which the head can turn should be such that the pupils in the eyes will still be visible. If the pupils disappear from the eyes, the face will look neither human nor believable.

The head can not only move in the horizontal and vertical direction and a combination of these, but there is also an element **<head-roll>** that makes it possible to move the head in an axial plane. This is essential for adding realism to the VH and is often used in conjunction with other elements, such as **<agree>** and other head movements.

Movements of the eyebrows are very common. At present there is only defined a vertical movement for the eyebrows, but an element for squeezing the eyebrows together, as for example when the face should look confused, would be profitable and therefore recommended to add to the language in the future.

When blinking, both eyes do not act exactly the same, one eye might start the blink before the other. This must be considered when implementing **<eye-blink>**. Though, the user would probably be confused if having to specify a start and end time for each eye in order to make it look natural. Therefore, these attributes do not exist, but this should still be taken care of in the implementation. Some blinks are double blinks, which means two quick blinks following on each other. Instead of using two different elements, which was the case in version 0.1 of VHML, one for single blinks and one for double blinks, a new attribute **repeat** was added to **<eye-blink>** to specify if the blink should be a single blink or of any other number. Since it is most common to do a single blink this will be the default value for the attribute. This attribute was also added to **<wink>**, since it should be possible to do several repeated winks. It was decided to keep **<wink>** rather than using **<eye-wink>** since a wink concerns more than the eye, for example the cheek.

Furthermore, **<left_wink>** and **<right_wink>** were merged to one **<wink>** element and given a **which** attribute to specify which side that should wink.

In version 0.3 of VHML, there is no way of moving the nose in any direction. However, an element that wrinkles the nose, by raising it, could be useful and should therefore be considered for future versions. This will not only include the nose, many other parts of the face will also be affected.

To make the names consistent the **<open-jaw>** and **<close-jaw>** elements were renamed to **<jaw-open>** and **<jaw-close>**, with the verb in the end. These elements can in the future be combined to constitute a yawn and thus be a part of the GML.

Figure 20 shows an example of how the facial animation elements can be used in a VHML document.

```
<?xml version="1.0"?>
<!DOCTYPE vhml SYSTEM "http://www.vhml.org/DTD/vhml.dtd">
<vhml>
   <p>
      <look-up intensity=”medium”>Look up there! I see a bird.</look-up>
      <eyes-left duration=”2500ms” intensity=”20”/>There is another one just
       next to it.<eye-blink duration=”100ms” repeat=”2”/>
   </p>
</vhml>
```

Figure 20. An example of a VHML document using facial animation elements.

3.7 Speech Markup Language

The *Speech Markup Language* (SML) only affects the voice of a VH, the face and body will not be affected. Table 12 shows a summary of the elements in SML. The emotions and gestures should also affect the voice and all those elements are therefore inherited from EML and GML.

Element	Description
break	Controls the pausing or other prosodic boundaries between words.
emphasise-syllable= emphasize-syllable	Emphasizes a syllable within a word.
phoneme	Provides a phonetic pronunciation for the contained text.
prosody	Controls the prosody of the contained text.
say-as	Controls the pronunciation of the contained text.
voice	Specifies the speaking voice of the contained text.

Table 12. A summary and description of the SML elements.

The first version of VHML had two elements to announce a break in an utterance, both **<break>** and **<pause>**. These were far too similar and therefore merged. The names of the element and the attributes were chosen with reference to SSML. The attribute **smooth** was kept from **<pause>** to make it possible to specify if the phoneme before the break should be lengthened slightly, even though SSML does not have a corresponding attribute.

In VHML Working Draft v. 0.1, there were two ways of emphasizing whole words or phrases and an additional element to emphasize syllables. In order to increase the simplicity of VHML, the two **<emphasis>** elements were merged into one element. This element was placed in GML, since it affects the face and body as well as the voice.

<emphasize-syllable> was kept from the earlier version of VHML in order to have a way of only emphasizing certain syllables in a word consisting of more than one syllable. This element has an attribute, **target**, to specify which syllable to emphasize. The element name can be spelled in two different ways since the word emphasize has different spellings for different English languages.

Instead of having one element for each prosody feature, for example one for pitch, one for rate and one for volume etc., all features are controlled by one element **<prosody>**. This element has consequently **pitch**, **rate**, **volume** etc. as attributes. By doing this change, the simplicity of VHML will increase without affecting the completeness. It is possible to give the attributes either a relative value, like +17%, or a descriptive value, like low, medium, high etc. This turned out to be a problem when specifying the DTD, since the only way to specify a relative value is by using CDATA, which allows all kinds of strings and thus also misspelled descriptive values. This is a situation when XML Schema would have been a better alternative, since it allows more specific type control.

To be compatible with SSML, **<say-as>** and **<emphasis-syllable>** were changed to use a hyphen instead of underline.

Although **<person>** sets the main characteristics of the voice, there is a need for a **<voice>** element to only change the voice of certain utterances. **<voice>** has the same attributes as **<person>**, apart from disposition. Some of the comparative languages used **<speaker>** as the name of this element, but since SSML uses **<voice>** and the element only affects the speech and not the face or body, **<voice>** was a more suitable name.

SSML uses **<audio>** and version 0.1 of VHML uses both **<audio>** and **<embed>** to include additional sounds to a document. Since **<embed>** allows other than just audio features, the **<embed>** has been retained and **<audio>** treated as a particular case of **<embed>**. It can occur anywhere in a document and was therefore placed at the top level of VHML. VoiceXML and Sable use attributes that add some special features to **<embed>**, like a way to specify if the audio should be played in the background or not. These features are not considered in the current version, but are recommended for future work.

Figure 21 shows an example on how to use the speech elements in a VHML document.

```
<?xml version="1.0"?>
<!DOCTYPE vhml SYSTEM "http://www.vhml.org/DTD/vhml.dtd">
<vhml>
   <person category=”adult” gender=”female”>
      <p>
         My son said his first word yesterday, which was
         <voice age=”2” gender=”male”>Mama</voice>
          Yesterday’s date was <say-as type=”date:md”> 3/1 </say-as>.
         <prosody rate=”fast”>When talking fast<break size=”small”/>it is
         important to include pauses.</prosody>
      </p>
   </person>
</vhml>
```

Figure 21. An example of a VHML document using speech elements.

3.8 Body Animation Markup Language

Although the *Body Animation Markup Language* (BAML) is the part of VHML taking care of the body animation, it has not been a part of this project. Therefore, there has been no change in BAML since the first version of VHML. However, since the emotions and gestures should affect the body, all EML and GML elements are inherited to BAML.

3.9 eXtensible HyperText Markup Language

The *eXtensible HyperText Markup Language* (XHTML) controls the text output from the application. The current version of VHML only constitutes a small subset of the existing XHTML, more precise only one single element. This is described in table 13.

Element	Description
anchor = a	Inserts an anchor in the output text.

Table 13. A summary and description of the XHTML element.

In VHML Working Draft v. 0.1, a much wider subset of elements was included, for example, different heading levels, bold, italics etc. These were affecting both the text output and the voice. To increase the simplicity of the language there should only be one way of changing the voice, i.e. using the SML elements. The other VHML elements can also be used to change the text output. The **<emphasis>** element can, for example, make the text italic or when speaking with a high volume the text can be capitalized or bold.

The only useful element that was not found any alternative VHML element for, was **<anchor>**. Therefore this was kept as the only XHTML element. This seemed to be an important feature of the language and was requested from the Interface group at Curtin.

For a person who is used to either XHTML or HTML, using **<a>** is the obvious way to insert an anchor in the text, though this is not very intuitive for a beginner. Therefore, both **<a>** and **<anchor>** can be used for denoting an anchor in the text. However, since a VHML document is an XML document one cannot blend **<a>** and **<anchor>**. The start and end elements have to be the same. To facilitate for advanced users used to XHTML or HTML, all original attributes to the **<a>** element are kept in the language.

Depending on the demands of the application this sub language might need to be expanded. There might for example be use of a way to specify that the text consist of code or lists, since these types of text should not be spoken in the same way as ordinary text. This can be done by using the **<code>** and **<pre>** elements defined in XHTML.

Figure 22 shows an example on how the anchor element with the **href** attribute can be used in a VHML document.

```
<?xml version="1.0"?>
<!DOCTYPE vhml SYSTEM "http://www.vhml.org/DTD/vhml.dtd">
<vhml>
   <p>
      Please, look for yourself and find out on
      <a href="http://www.vhml.org">the VHML web page</a>.
   </p>
</vhml>
```

Figure 22. An example of a VHML document using the XHTML element.

3.10 Dialogue Manager Markup Language

The *Dialogue Manager Markup Language* (DMML) has not been refined as a part of this project and hence will not be described in this thesis.

3.11 Discussion

Many changes have been made from the first version of VHML to fulfil the criteria for a stable markup language. All these changes have resulted in a third version of the

VHML Working Draft (VHML v. 0.3, 2001). The work with this language does not end here. VHML will successively be improved and new versions of the specification will appear.

There are many features of VHML that have been considered, but are not yet added to the language. The reason for this is that not enough investigation about these features has been made because of the time constraints of this project.

- When XML Schema has become more stable and there are free parsers to download, it might be an advantage to change from using DTD to use schema in order to get all the extra features given by schemas.
- The speaker of a document is defined by specifying age, category and gender, but many additional properties, for example nationality/culture, physique etc. might affect how the VH acts in terms of the face, body and voice. The properties may be added as attributes to **`<person>`** and maybe even to the **`<voice>`** element. Which set of properties that is profitable when developing a VH has not been investigated.
- There are only two types of files that are possible to embed within a VHML document, AU and MML files, but many other file types can be of interest. As this depends on the requirements for each separate application, this project has not considered which file types that will be required or of benefit for being able to embed.
- There are nine different emotions that can be used for a VH. However, there are a very large number of emotions to choose from, some more common and unambiguously expressed than others. Which emotions that are meaningful to include in the language is a big issue of research. Nevertheless, some of the emotions that already exist in VHML only affect the face. Investigation on how these emotions affect the body and voice is required to be able to define and implement them.
- A way of producing new emotions, without specifying them as new elements in the language, is to blend already existing emotions. How this should be done and which attributes that are required for the emotions in order to do so, have not been investigated.
- In the same way as for emotions, there are many gestures that may be added to GML. These could for example be think, whistle, yawn etc. It must be carefully investigated how a person acts in terms of the face, body and voice when doing different gestures. Some of the already existing gestures only affect the face, but should also be defined for the body as well as the voice. This as well has to be carefully investigated in order to define and implement them.
- Since SML is based on SSML, XML Namespaces could be used to inherit the exact elements. The advantage of this is that if SSML changes, these changes will have effect on VHML as well. What has to be taken into consideration though, is that some of the VHML elements have additional attributes, which do not exist in the SSML elements and this is a problem that has to be solved. The reason why namespaces is not used in this version of VHML is that SSML so far is only a working draft, which means that the SSML elements do not exist in the way that they can be inherited by using namespaces. When SSML becomes a standard, the elements might have been slightly changed, which

possibly will affect VHML. Therefore, another version of VHML should be developed when SSML becomes standardized.

- There are a limited number of movements that are possible to be expressed in the face of a VH. Therefore, it can be profitable to define more movements, for example, a way of raising the nose in order to wrinkle it or squeezing the eyebrows together. In order to simulate other movements, a lower abstraction level is used, like changing the FAPs as in MPEG-4.
- Nothing is specified in VHML for the body movements, but at present, a group in Switzerland that is a part of InterFace, is researching on this part of a VH. Taking advantage of their expertise would be profitable when defining BAML.

4 Dialogue Management Tool

In order to create a useful tool for constructing and maintaining the type of dialogues that were described in section 2.8, the *Dialogue Management Tool* (DMT) has been designed, implemented, tested and informally evaluated. The DMT makes construction of dialogues easier and keeps track of the state traversing in a conversation. Currently the DMT is based on responses marked up in VHML. This version of the DMT has been found adequate for developing three other applications, the *Mentor System* developed by Marriott (to be published), the *FAQBot* developed by Beard (1999) and *The Mystery at West Bay Hospital*, section 5.3.

4.1 Dialogue Management Tool Language

The main objective of the DMT is that it should be a useful tool when creating and maintaining dialogues. These dialogues can be included when developing, for example, an interactive Talking Head application or when planning an ordinary question and answer file. It uses the XML-based markup language *Dialogue Management Tool Language* (DMTL), developed within this project, to represent the dialogue and its states as a network. The DTD for DMTL can be found in Appendix D. The overall structure of DMTL is shown in figure 23. An example of how DMTL can be used can be found in section 4.1.12.

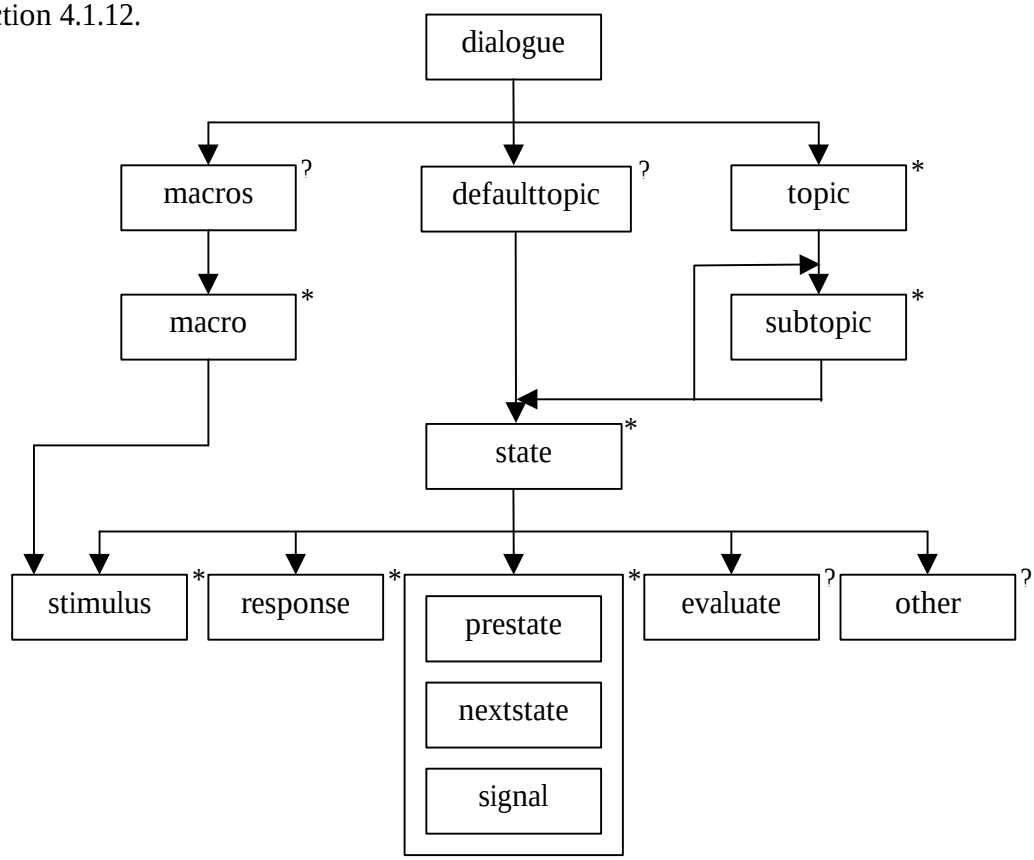


Figure 23. The structure of DMTL.

In figure 23, an arrow from A to B means that A can consist of B. The number of B's is specified using stars and question marks. A star (*) after the element means that the element can occur zero or more times and a question mark (?) that the element can

occur zero or one time. A summary of the elements and their attributes are presented in table 14.

Element	Attributes	Contains
dialogue	-	macros? defaulttopic? topic*
macros	-	macro*
macro	name	stimulus*
defaulttopic	-	state*
topic	name	subtopic*
subtopic	name keywords evaluate	subtopic* state*
state	name type	stimulus* response* prestate* nextstate* signal* evaluate? other?
stimulus	type	character data
response	weight statereference	character data
prestate	name	empty
nextstate	name	empty
signal	name	empty
evaluate	-	character data
other	-	character data

Table 14. DMTL elements.

DMTL has been developed in close cooperation with the Interface group at Curtin and therefore it is known that DMTL offers all currently desirable functionality. The design of DMTL was tied to the Curtin requirements and future applications may require alterations to the language.

4.1.1 Dialogue

The root element in DMTL is **<dialogue>**. This can include zero or one **<macros>**, zero or one **<defaulttopic>** and zero or more **<topic>** elements.

```
<dialogue>
   <macros>…</macros>
   <defaulttopic>…</defaulttopic>
   <topic name="greeting">…</topic>
   <topic name="VHML">…</topic>
   …
</dialogue>
```

4.1.2 Macros

The **<macros>** element includes zero or more **<macro>** elements, which in turn includes zero or more **<stimulus>** elements, section 4.1.7. **<macros>** was introduced to DMTL in order to make it easier for the user of the DMT when creating stimuli. When creating stimuli, all different ways of giving a specific stimulus must be considered. Since natural language is complex, there are many different ways to express the same question. **<macros>** can be created to match the semantic of a certain stimulus.

For example, the macro "WHATIS" can be used in the sentence "WHATIS VHML?" within a stimulus, this is shown in the example given in section 4.1.12. This is then defined to match "What is VHML?", "Can you please tell me about VHML." and so on. In order to differ from ordinary text in the stimulus, the macro names are in capital letters.

```
<macros>
   <macro name="WHATIS">
      <stimulus>What is</stimulus>
      <stimulus>Can you please tell me about</stimulus>
      …
   </macro>
   …
</macros>
```

4.1.3 Defaulttopic

The **<defaulttopic>** caters for all the user input that does not match any other **<stimulus>**, section 4.1.7. The **<defaulttopic>** can contain zero or more **<state>** elements, section 4.1.6, and hence gives the user a possibility to have many different default responses. This can be useful when having responses such as "Sorry, but I can't understand that." or "Sorry, I can't help you with that.". The idea with **<defaulttopic>** is to give the user a possibility to design these default responses in a specific way, best suitable for their specific application.

```
<defaulttopic>
   <state name="default1">…</state>
   <state name="default2">…</state>
   …
</defaulttopic >
```

4.1.4 Topic

A **<topic>** includes zero or more **<subtopic>** elements. A **<topic>** has a required attribute **name** that is an identifier for the **<topic>**. By using **<topic>** elements the structure of the dialogue becomes organized and well presented.

```
<topic name="VHML">
   <subtopic name="whatis">…</subtopic>
   <subtopic name="dtd">…</subtopic>
   …
</topic>
```

4.1.5 Subtopic

A **<subtopic>** in turn includes zero or more **<subtopic>** elements and zero or more **<state>** elements. Also the **<subtopic>** has a required attribute **name** as an identifier.

```
<subtopic name="whatis">
   <subtopic name="question">…</subtopic>
   …
   <state name="name">…</state>
   …
</subtopic>
```

Dialogues tend to grow fast and become large and complex, with many topics, subtopics and states. This becomes an efficiency problem when a *Dialogue Manager* (DM) has to parse all the different paths in the dialogue when searching for a suitable stimulus. To avoid this, an attribute, **keywords**, for the **<subtopic>** element was introduced. This makes it possible to specify a number of keywords for each subtopic

and only if any of these match the user input the subtopic is parsed to find a suitable state. If no keywords are specified for a subtopic no shortcut is provided and the DM must perform a full search.

```
<subtopic name="whatis" keywords="vhml,about">
   …
</subtopic>
```

If this subtopic is to be parsed, the user input must at least match one of the keywords *vhml* or *about*.

Yet another way to decrease the numbers of paths to parse is to use the **evaluate** attribute for the **<subtopic>** element. With **evaluate**, some conditions can be set and these have to be fulfilled in order to parse that specific subtopic.

```
<subtopic name="whatis"
          evaluate="-test State_VHML.whatis.name_visited">
   …
</subtopic>
```

In order to parse this subtopic, the evaluate condition must be *true*. In this example, the condition is true if the state *VHML.whatis.name* has been visited before in the dialogue. However, what values **evaluate** can have is up to the DM that parses the DMTL file to specify.

4.1.6 State

A **<state>** includes **<stimulus>**, **<response>**, **<prestate>**, **<nextstate>**, **<signal>**, **<evaluate>** and **<other>**. A **<state>** has an attribute, **name**, that works as an identifier for the specific **<state>**.

In the current version of DMTL there are four different values for the **<state>** attribute **type**:

- **active**. A state that invokes a question, without having to be triggered by a stimulus. For example, the question "Do you want to know more about VHML?".
- **entry**. A state that can be invoked at any time during the dialogue if the stimulus matches. This is also the default state type. An example of this is the user input "What is VHML?".
- **linked**. A state that is connected to other states by using **<nextstate>** or **<prestate>**. The state is linked because the stimulus depends on having some kind of context to be understood correctly. An example is the user input "What is that?", where "that" corresponds to something introduced earlier in the conversation and the DM should know what it is. A **linked** state can never directly match the initial user input, it has to be linked from another state.
- **visitswitch**. A state that points to several other states and works in a similar way as a case statement in C or Java. The state the dialogue moves into can, for example, depend on whether the state has been visited before. The **visitswitch** specifies the priority order in which the states should be moved into, but makes certain that no state is visited more than once. An example on where to use the **visitswitch** is if the user types in "Can you tell me about VHML". If it is the first time this question is asked the **visitswitch** can point to a certain answer "Have you tried to look at the VHML web page".

However, the next time the same question is asked, the user does not want the same answer and the **visitswitch** can direct the answer to contribute with something new to the user, like "You can read the VHML specification on the VHML web page".

Examples on how the different types are used are given in section 4.1.12.

4.1.7 Stimulus

The **<state>** can have zero or more **<stimulus>** elements. A **<stimulus>** can be of four different **types** depending on the application; **text**, **audio**, **visual** and **haptic**, with **text** as the default value. For example, instead of having "Yes" as a text stimulus, there can be a visual stimulus that is triggered when the user nods. This is represented with "usernod" in the following example:

```
<state name="agree">
   <stimulus>Ok.</stimulus>
   <stimulus type="text">Yes.</stimulus>
   <stimulus type="visual">usernod</stimulus>
   …
</state>
```

In this example, "Ok." has not got a value for the **type** attribute and hence gets the default value **text**.

4.1.8 Response

The **<state>** can have zero or more **<response>** elements. A **<response>** could be plain text or marked up in any language. For example, the question and answer structure in a FAQ file could be maintained by using just the stimuli and responses. The **<response>** could also be marked up to direct or control the way in which the response is presented, for example, by using HTML anchors.

Further, the **<response>** has an attribute **weight** with the default value "0.7". This can be used by the DM when there exists more than one response and it has to be decided which one to use in the application. This gives the user a possibility to specify the preferred response to the DM. If there is more than one response with the same weight, the DM can randomly choose which one to use. This enables the TH to be more varied.

```
<state name="agree">
   …
   <response>Then I will tell you about it…</response>
   <response weight="0.8">
      Ok. Let me explain that to you…
   </response>
   …
</state>
```

In this example, the response beginning with "Then I will tell you about it…" does not have a value for the **weight** attribute and hence gets the default value "0.7".

Another attribute, **statereference**, was added to the **<response>** element to make it possible for two different states to have the same responses. This is a useful feature when, for example, the user asks a question like "What is VHML?" or if the user previously has been introduced to the concept VHML and asks: "What is that?". These two questions should trigger the same responses, but the first one has to be an **entry**

state and the second one a **linked** state. This since the first question can be posed during any time in the dialogue and the other question must have a context where "that" refers to something that has been introduced earlier. To avoid having to type in the same responses twice or even more, the **statereference** can be used. A response that specifies a **statereference** has exactly the same responses as the referred state has and hence can not have any additional responses. This can not be controlled within the DMTL DTD, but a check is made in the DMT.

```
<subtopic name="whatis">
   <state name="name" type="entry">
      <stimulus>WHATIS VHML?</stimulus>
      …
      <response>
            VHML is a markup language for Virtual Humans.
      </response>
      …
   </state>
   <state name="pronoun" type="linked">
      <stimulus>WHATIS that?</stimulus>
      …
      <response statereference="VHML.whatis.name"/>
      …
   </state>
   …
</subtopic>
```

The second state, *pronoun*, has no responses but has a **statereference** pointing to the state *name* and hence has the same responses as the specified reference. The **statereference** is in a specific format called *fully qualified names*, section 4.4.1.

4.1.9 Prestate, nextstate and signal

The **<state>** can also contain zero or more **<prestate>**, **<nextstate>** and **<signal>** elements. These can appear in the state in any order to make it easier for a user that does not use the DMT, but is constructing their dialogue in an ordinary text editor. In this way, the user does not have to remember in which order they have to appear, just the correct element names. The DMT inserts the element in the following order: **<prestate>**, **<nextstate>** and **<signal>**.

<prestate> specifies the states from which the dialogue could have come and **<nextstate>** the states to which the dialogue can move. There was a considerable debate on whether or not both **<prestate>** and **<nextstate>** should remain in DMTL, but it was decided to keep both because it gives the user an opportunity to chose which one to use. There is no difference in functionality between the two within the DMTL. What can be done with one, can also be done with the other. The only difference is the element name, but they represent different views of how a dialogue is structured. It is up to the user and the DM to choose whether to use one or the other or even both.

An example of how to use **<nextstate>** is:

```
<subtopic name="question">
   <state name="about" type="active">
      <response>Do you want to know more about VHML?</response>
      <nextstate name="VHML.whatis.question.agree"/>
      <nextstate name="VHML.whatis.question.disagree"/>
   </state>
   …
```

```
      <state name="agree" type="linked">
         <stimulus>Ok.</stimulus>
         <stimulus type="text">Yes.</stimulus>
         <stimulus type="visual">usernod</stimulus>
         <response>Then I will tell you about it…</response>
         <response weight="0.8">
               Ok. Let me explain that to you…
         </response>
      </state>
      <state name="disagree" type="linked">…</state>
   </subtopic>
```

Here **<nextstate>** is used to indicate that the *agree* and *disagree* states can follow from the *about* state. Also, the *agree* and *disagree* states are linked states and hence can only be moved into from another state. The **<nextstate>** is specified with *fully qualified names*, section 4.4.1.

The **<signal>** element enables the match to generate or emit a signal or notification to the DM, which it may choose to ignore or handle in some way. For example, if the user says "Good bye" the DM may choose to close the connection. What values **<signal>** can have is up to the DM to decide.

```
   <state name="goodbye" type="entry">
      <stimulus>Good bye.</stimulus>
      <signal name="exit"/>
   </state>
```

4.1.10 Evaluate

A **<state>** can have zero or one **<evaluate>** element. The **<evaluate>** element can be used for defining a condition that has to be fulfilled before the dialogue is able to move into this particular state. For example, a variable can be set to imply that a state is visited.

```
   <state name="name" type="entry">
      <stimulus>WHATIS VHML?</stimulus>
      …
      <evaluate>-visited State_name</evaluate>
      …
   </state>
```

4.1.11 Other

A **<state>** can have zero or one **<other>** element. **<other>** can be used for specifying any additional application specific information necessary or simply to add comments about the state.

```
   <state name="name" type="entry">
      <stimulus>WHATIS VHML?</stimulus>
      …
      <other>Information about VHML</other>
   </state>
```

4.1.12 DMTL example

Everything that has been explained so far is gathered in a fragment of one single dialogue:

```
<dialogue>
   <macros>
      <macro name="WHATIS">
         <stimulus>What is</stimulus>
         <stimulus>Can you please tell me about</stimulus>
         …
      </macro>
      …
   </macros>
   <defaulttopic>
      <state name="default1">…</state>
      <state name="default2">…</state>
      …
   </defaulttopic>
   <topic name="greeting">
      <subtopic="endphrase">
         <state name="goodbye" type="entry">
            <stimulus>Good bye.</stimulus>
            <signal name="exit"/>
         </state>
         …
      </subtopic>
      …
   </topic>
   <topic name="VHML">
      <subtopic name="whatis">
         <subtopic name="question">…</subtopic>
            <state name="about" type="active">
               <response>
                  Do you want to know more about VHML?
               </response>
               <nextstate name="VHML.whatis.question.agree"/>
               <nextstate name="VHML.whatis.question.disagree"/>
            </state>
            <state name="agree" type="linked">
               <stimulus>Ok.</stimulus>
               <stimulus type="text">Yes.</stimulus>
               <stimulus type="visual">usernod</stimulus>
               <response>
                  Then I will tell you about it…
               </response>
               <response weight="0.8">
                  Ok. Let me explain that to you…
               </response>
            </state>
            <state name="disagree" type="linked">…</state>
         </subtopic>
         <state name="name" type="entry">
            <stimulus>WHATIS VHML?</stimulus>
            <response>
               VHML is a markup language for Virtual Humans.
            </response>
            <evaluate>-visited State_VHML.whatis.name</evaluate>
            <other>Information about VHML</other>
         </state>
         <state name="pronoun" type="linked">
            <stimulus>WHATIS that?</stimulus>
            <response statereference="VHML.whatis.name"/>
         </state>
         …
      </subtopic>
```

```
        <subtopic name="dtd">…</subtopic>
        …
    </topic>
    …
</dialogue>
```

Examples of DMTL files can be found at: http://www.vhml.org/downloads/DMT.

4.2 Requirements

The requirements of the DMT application were divided into two different levels; basic and future work. The contents of the basic were functions that was to be implemented and completed during this project. The contents of future work was not considered, even though preparations for some of these functions were included in the *Graphical User Interface* (GUI) but were shadowed to show that there is no current implementation. Much of the future work was directed towards the requirement of the *Mentor System* developed by Marriott (to be published).

4.2.1 Open file

Basic

It should be possible to either create a new dialogue file or open an already existing dialogue file. Existing files must be valid DMTL documents, if not, an error message should be presented to the user.

Future work

When opening a new or an existing file the user should be able to choose between different file types, for example, DMTL file, Question/Answer file, Text file, Mentor topic entity file and Metaface topic entity file. Other file types may be of interest as well.

4.2.2 Save file

Basic

It should be possible to save and name an unnamed file by specifying a name and the path to the directory as well as save and rename an already named file.

4.2.3 Import file

Future work

The user should be able to write a Question/Answer file, Text file, Mentor topic entity file or a Metaface topic entity file in any editor and then import the file into the DMT. Other file types may be of interest as well.

4.2.4 Export file

Future work

The user should be able to export the viewed DMTL file by transforming it to a Question/Answer file, Text file, Mentor topic entity file or a Metaface topic entity file. Other file types may be of interest as well.

4.2.5 Print file

Future work

It should be possible to convert a viewed file to either PostScript or HTML format.

The user should be able to choose what parts to be printed. The targets are so far specified to suit the *Mentor System* (Marriott, to be published); i.e. current list, entire list, current active list or entire active list.

4.2.6 Quit DMT

Basic

The user should be able to quit the application whenever wanted. If the viewed file has unsaved changes it should be possible to quit without saving, save and then quit or cancel the action and return to the application.

4.2.7 Edit

Basic

It should be possible to add all the state elements, these are **<stimulus>**, **<response>**, **<prestate>**, **<nextstate>**, **<signal>**, **<evaluate>** and **<other>**.

When editing **<stimulus>** and **<response>**, the user should be able to either type directly in the stimuli and responses areas in the DMT or in an editor called GVim. If GVim is preferred, the user should be able to choose to open the editor to write either stimuli or responses. The file opened in GVim should then contain the information from the stimuli or responses area, if any exists. After finished typing in the editor, the file has to be loaded into the DMT in order to be included in the viewed dialogue.

If the user chooses to type in the specified area for responses in the DMT, there should be a number of predefined functions to use for making the editing more convenient. These functions should be developed to suit creating a VHML dialogue, since VHML can be useful when controlling the output of a TH or a VH application and is a significant part of this project.

The user should be able to undo recently made changes regarding **<stimulus>**, **<response>**, **<prestate>**, **<nextstate>**, **<signal>**, **<evaluate>** or **<other>** within the viewed **<state>**. It should be possible to undo more than just the last change.

The user should be able to redo changes that have been undone regarding **<stimulus>**, **<response>**, **<prestate>**, **<nextstate>**, **<signal>**, **<evaluate>** or **<other>** within the viewed **<state>**. It should be possible to redo more than just the last change that has been undone.

The user should be able to create **<macro>** elements with a specific **name**. The user should also be able to rename an existing **<macro>**, edit a **<macro>** by editing **<stimulus>** elements or delete a **<macro>**.

The user should be able to create a new **<state>** in the **<defaulttopic>**, including specifying a **name**. The user should be able to view the default states and edit the states in the same way as any other state.

The user should be able to create a new **<topic>**, including specifying a **name**. The new **<topic>** should be included in the viewed dialogue. The user should also be able to

rename an existing **<topic>**, edit a **<topic>** by adding **<subtopic>** elements or delete a **<topic>**.

The user should be able to create a new **<subtopic>**, including specifying **name**, **evaluate** and **keywords**. The new **<subtopic>** should be included in the viewed dialogue. The user should also be able to rename an existing **<subtopic>**, edit a **<subtopic>** by editing the **keywords** or **evaluate** or by adding **<subtopic>** and **<state>** elements or delete a **<subtopic>**.

It should also be possible to view a **<subtopic>**. The **<state>** elements included in that particular **<subtopic>** should be presented.

The user should be able to create a new **<state>** to a specific **<subtopic>**, by specifying a **name** and selecting the correct **type** of the **<state>**, i.e. **active**, **entry**, **visitswitch** or **linked**. The new **<state>** should be included in the viewed **<subtopic>**.

By selecting a certain **<state>**, the user should be able to view and edit the **<stimulus>**, **<response>**, **<prestate>**, **<nextstate>**, **<signal>**, **<evaluate>** or **<other>** that correspond to that particular **<state>**. It should also be possible to delete or rename a **<state>** and change the type of the **<state>**.

Future work

The user should be able to edit **<stimulus>** and **<response>** in any editor, not just GVim, and then load this file into the DMT.

The predefined functions connected to the **<response>** text area should be written in the user's language of choice.

It should be possible to reorder a dialogue by cutting, copying and pasting any object in the application, for example a **<state>** or a reference in **<nextstate>**.

It should be possible to undo and redo any action made in the application.

4.2.8 View

Future work

The user should be able to view the selected **<subtopic>** in different ways, i.e. current list, entire list, current active list and entire active list. The targets are so far specified to suit the *Mentor System* (Marriott, to be published).

4.2.9 Options

Future work

The user should be able to choose between showing a brief or a long description of the **<state>** elements.

4.2.10 Help

Basic

The user should be able to obtain on-line help concerning the functions of the DMT and a short summary of the application.

The user should also get warnings or error messages as soon as an error has occurred. These messages should disappear as soon as the next correct action is performed.

4.3 Implementation

The implementation of the DMT was made in Java 1.3.1 and was documented with JavaDoc v. 1.3.

4.3.1 DOM tree

A decision had to be taken whether to use a DOM API or a SAX API for processing the DMTL document. Since the whole tree had to be kept in memory at once in order to be able to make changes in the tree, the DOM API was considered to be the best alternative. The reason for this is that a DOM tree, section 2.6.4, allows the user to go back and forth in the document, whilst a SAX API forces the user to make the changes immediately.

Input to the DMT is stored as a DOM tree and saved as a DMTL document. The DOM tree updates dynamically when the user makes changes, via an auto save routine. The tree is not printed to file and saved as a DMTL document until the user actively chooses to do so by selecting the save function. This file constitutes a static status of the new, updated DOM tree.

In order to find the right state to make changes to, pointers to the states in the viewed subtopic are stored in an array. When that subtopic is chosen, its states are presented on the screen as a list. The index for a specific state in the list corresponds to the index in the array of state pointers. In this way, not every state has to be searched to find the one that is to be changed. The correct state is picked out from the array using the index number selected on the screen.

4.3.2 The Graphical User Interface

The DMT *Graphical User Interface* (GUI) is shown in figure 24. It has been developed on the basis of the *Mentor System* (Marriott, to be published). A detailed description of the user interface, as well as the functionality, can be found in the user manual, Appendix E.

During the development of the GUI a number of criteria were defined and taken into consideration.

- *Simplicity*. The GUI should not look complicated. For example, the colours should be distinct and the images clear. Similar functions should be grouped together and it should be obvious which functions that can be used for each situation.
- *Consistency*. Terms and images used in the GUI should be consistent, both within the GUI and regarding other existing user interfaces, for example, words should be in the same form.
- *Intuitivity*. The position of each component should be intuitive as well as terms and images used in the GUI. They should clearly describe their functionality.
- *Usability*. There should be features in the GUI that suit both beginners and advanced users. This can be achieved by including different types of shortcuts.

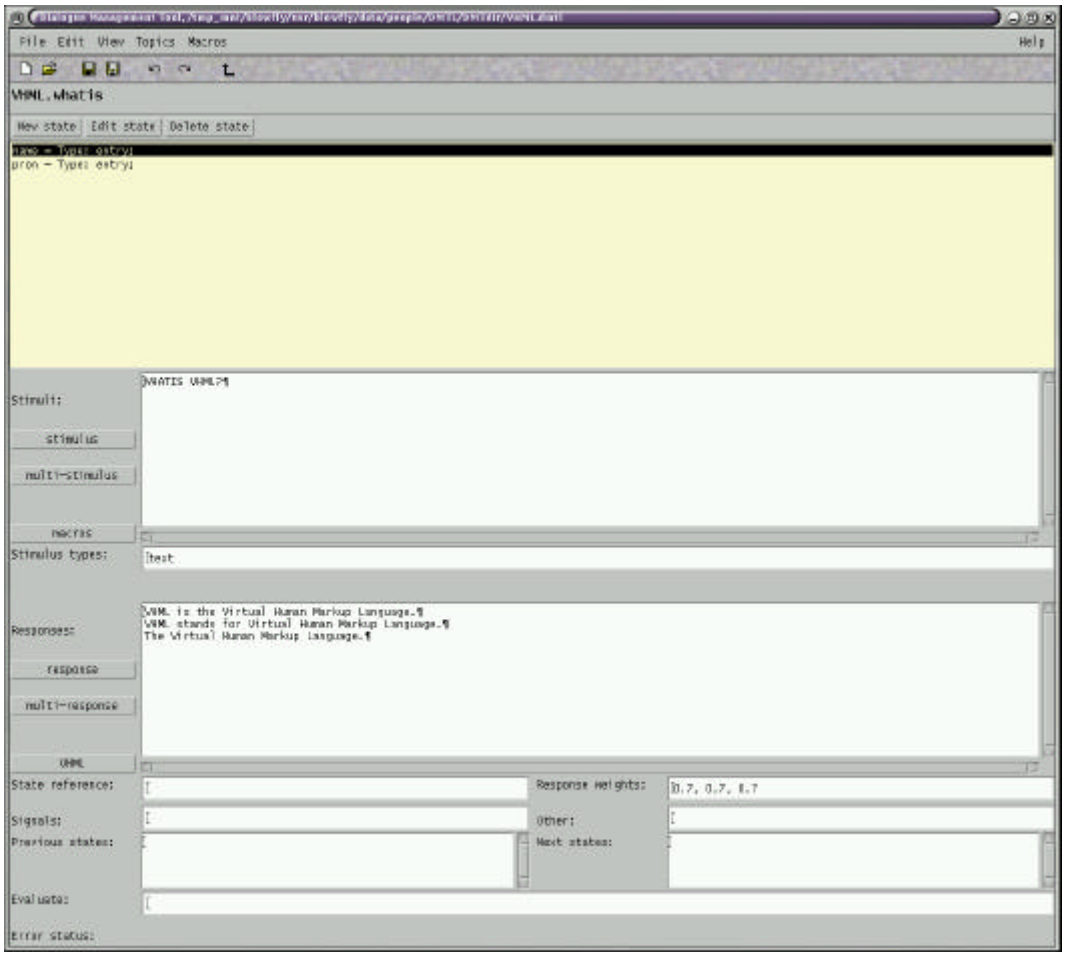


Figure 24. The DMT GUI.

4.4 Problems

During the implementation, some problems have arisen. Firstly, there was a problem keeping track of which node in the DOM tree that was active. Secondly, there was a problem when having XML-based elements inside the response elements. A third problem was to print out the dialogue to a DMTL document that should be readable by humans and not just machines.

4.4.1 Fully qualified names

To keep track of which state is active, *fully qualified names* are used. A fully qualified name is a name that gives the whole search path from the root element. For example, a state called *name* in a subtopic *whatis* in a topic *VHML*, has the fully qualified name *VHML.whatis.name.* By using these names it is possible to keep track of in which path in the DOM tree the active state is situated and changes can easily be made inside that particular state.

The same technique is used when a state is referred to in a **<nextstate>** or in a **<prestate>**. This made it possible to refer to states in other subtopics or even in other topics.

Also in the attribute **statereference** inside the **<response>** element, it is possible to refer to states in other subtopics or topics. By using fully qualified names, when specifying a state as a **statereference**, the updating of the responses becomes easier.

The user does not have to look through every state when one response is changing, just the one that originally contains the response.

If a topic, subtopic or state is renamed, the DMT finds all references to that element and replaces the reference with the new one. The same thing occurs when a topic, subtopic or state is deleted. All references to this element are then removed. This keeps the dialogue stable and makes sure there are no references to non-existing states.

Another advantage of having fully qualified names is to prevent the user from editing references to non-existing states. However, having to type the fully qualified name when the state is situated in the same subtopic as the referring state is time inefficient. A solution to this problem could be to let the DMT use scoping, i.e. that a name is defined in the element itself, but also in any elements within that element. Because of the time constraints in the project this has not been investigated further.

4.4.2 XML-based

The responses in the dialogues may be marked up in an XML-based language, for example VHML. To include other XML elements inside the **<response>** elements will cause problems. Because these elements are not, and should not be, included in the DMTL DTD, the DMTL document will not be valid if they remain inside the responses. The solution to this was to implement a transform function that transforms the elements into plain text by using the standard entities for XML, section 2.6.1.

The following example includes responses marked up in VHML:

```
<response>
   <vhml>
      <p>
         <happy intensity="90">
            I am feeling happy today!
         </happy>
      </p>
   </vhml>
</response>
```

This is transformed into:

```
<response>
   &lt;vhml&gt;
      &lt;p&gt;
         &lt;happy intensity=&quote;90&quote;&gt;
            I am feeling happy today!
         &lt;/happy&gt;
      &lt;/p&gt;
   &lt;/vhml&gt;
</response>
```

Another problem is that, inside the **<vhml>** element these standard entities may already be used. If, for example, an apostrophe is needed in the response, the user has to type in the standard entity ' instead of the character ', as in any other XML document. The ' is then transformed into plain text, i.e. '. This can be used when the response, for example, includes "I'm" instead of "I am":

```
<response>
   <vhml>
      <p>
         <happy intensity=”90”>
            I&apos;m feeling happy today!
         </happy>
      </p>
   </vhml>
</response>
```

This transformed into:

```
<response>
   &lt;vhml&gt;
      &lt;p&gt;
         &lt;happy intensity=&quote;90&quote;&gt;
            I&amp;apos;m feeling happy today!
         &lt;/happy&gt;
      &lt;/p&gt;
   &lt;/vhml&gt;
</response>
```

4.4.3 Print to file

To make the DMT useful, it is important for the output from the application to be readable by humans. In this way, the DMTL files can be constructed and maintained both with and without the DMT. When writing a DMTL file without using the DMT, the easiest way is to use indentation to keep track of on what level topics, subtopics and states appear. Thus, when saving a dialogue as a DMTL file, DMT uses indentation. Further, the DMT reorders elements in the state into the preferred order, i.e. **<prestate>**, **<nextstate>**, **<signal>**.

4.5 Testing

All basic requirements of the DMT, section 4.2, were achieved. The tests were carried out by two different testers using a test schedule, Appendix F. The testers have not been involved in the implementation of the DMT and can therefore been seen as objective testers. The testing was continued until no more errors were found in the application, which turned out to be eight times. After each test round, the errors found in the DMT were corrected and a new version was released. The results from the eight different test rounds are summarized in table 15. The testers' duplicate errors are excluded.

Test round	Minor errors	Large errors	Total errors
1	30	23	53
2	13	17	30
3	4	8	12
4	2	5	7
5	3	3	6
6	-	3	3
7	1	4	5
8	-	-	-
Total	53	63	116

Table 15. Summary of the test results.

The errors are divided into two different levels; minor errors and large errors. The minor errors are mainly errors concerning the GUI. These include, for example:

- Shadowing menu items, labels and buttons that do not provide any functionality for this version of the DMT.
- Shadowing menu items, labels and buttons that cannot be used in a specific situation. For example, a new state cannot be created before a subtopic is selected.
- The consistency and correctness in the warning and error messages.
- Misspelling and grammatical errors in the GUI.
- Where to place the marker after an action has been performed.

The large ones include for example:

- Removing all references to a state that is deleted.
- Prevent the possibility to create topics, subtopics and states with no name.
- Prevent information from disappearing when new values were entered.

The number of errors decreases with every test round, except for round seven. In the first two rounds there were many errors, especially minor ones due to the time constraints of the implementation and since the DMT had not been tested completely by the programmer before the application was released for testing. Though, the number of errors decreased significantly and in the final round none were found.

4.6 How to use the system

Besides the DMT itself, a user manual and a guide for future programmers have been created.

The user manual can be found as Appendix E and can also be downloaded at http://www.vhml.org/documents/DMT/. It includes a description of the application as well as hints for the user.

To make the maintenance and further development of the DMT as easy as possible, the code for the DMT is well documented using JavaDoc v. 1.3. That documentation can be found at http://www.vhml.org/downloads/DMT/. It is highly recommended that future programmers read sections 4.7 and 6.2 in order to get an overview of what has been done and what should be further investigated.

4.7 Discussion

Several improvements can be made to the DMT. Some are requirements, but were considered future work, section 4.2, some were discovered during the development of the DMT. These have not been considered for this version of the DMT because of the time limit of the project.

- The DMT should neither let the user enter a name to a non-existing state nor force the user to type in the whole *fully qualified name* if this is not necessary. *Scoping* might solve this problem and hence has to be investigated.
- In the current version of DMT, it is not possible to cut, copy and paste any elements using the GUI. This is a feature that might be useful, so that the user can reorganize the dialogue if needed.

- There has not been any work done regarding importing and exporting DMTL files from and to other file types. Both the technique behind the import and export as well as what file types that should be considered has to be investigated.
- Further, this version of the DMT was developed to suit responses marked up in VHML. There might be other markup languages for which DMT may provide useful support, which ones have not yet been investigated.

5 Talking Head application

During the project, VHML has been validated and verified and then converted to an XML-based language, section 1. In the second part of the project, a language (DMTL) and a tool (DMT) for creating dialogues that can be used in the development of interactive TH applications were designed and implemented, section 4. The objective of the third part of the project was to create interactive TH applications in order to demonstrate the new VHML and the DMT.

The Interface group at Curtin has developed TH applications since 1992. For example, an adventure game application, which is based on the work done by Huynh (2000) and by Stallo (2000), was marked up with FAML and SML. The project group exhibited this game at a small, informal science fair in order to gain a preliminary evaluation of people's reaction to TH applications.

5.1 Initial evaluation

In August the 31st, the project group was involved with the TripleS Science Fair, a fair that shows cases of a number of different types of science research. The target group of the fair is children from the age of seven up to high school level and their families. The School of Computing at Curtin presented a TH application, the *Adventure Game*. The *Adventure Game* is an interactive story that changes direction depending on the user input. To reach the goal of the game, one has to walk to the right locations, pick up certain items, use the items in appropriate situations and at the end solve a riddle. The application includes both a TH and the text being spoken, which appears next to the TH. The text is marked up with FAML elements for expressing emotions in the face and SML elements for expressing emotions in the text spoken by the TH. It should be pointed out that these versions of FAML (Huynh, 2000) and SML (Stallo, 2000) are the original ones, not the ones included in VHML v. 0.4.

The presentation included three computers with the *Adventure Game* on each machine, but with different TH models. Two of the THs were realistic heads, built on two different pictures, *John* and *Bernie*. The third TH, *Loris*, was not realistic since the colours of the face did not look human. The three machines were placed pretty close to each other and the middle one, with *Bernie* as a TH, was connected to a projector.

5.1.1 Preparation

Since the aim of the fair was to present different types of science, including a presentation of computer science, it was not an ideal place for performing a big evaluation and thus no questionnaire was created. Instead, the users were supervised during the game and short conversations were held with the users, trying to get their overall opinion of the application.

Before going to the fair, the *Adventure Game* was played by the project group. The overall impression was that the game was quite boring. There was a very long introduction that did not require any interaction at all from the user. On the whole, there were very long intervals between the situations in the game where interaction from the user was needed and this was not appreciated. Another drawback was when one had made a decision about which action to take, one had to wait until the TH had finished speaking before giving the command for that action. Because of these

disadvantages, the expectation was that the game would not be very popular at the science fair.

The aims of the supervision and the conversations with the users were the following:

- Get an impression of what the user thought of the game itself; if it was fun, boring and so on.
- Find out if the time between the interactions from the user was adequate.
- Find out what they thought of the TH, concerning look, sound and usefulness.
- Understand whether the user only read the text, only looked at and listened to the head or did both.
- Catch if the user understood what emotions the TH was expressing.

5.1.2 Discussion

When analysing the results from the evaluation, one has to take into consideration that there was a fairly small number of people that was observed, approximately thirty, that it was not a controlled environment and that several factors may have distracted the user in different ways. In addition to this, there was a bug in the application that made it shut down if a certain action was performed. This may have caused some of the users to give up their attempts to complete the game. But on the other hand, it showed that some of the users were so interested in the game that they started all over again, even if the application shut down because of the bug. Another important issue to take into consideration is that people who come to a science fair probably have some sort of interest in science and therefore cannot be seen as randomly picked users. Further, the ones who ended up trying the game might have been the ones with most computer experience. Therefore, the result cannot be seen as proof in any way, just as an indication of which direction the development of this kind of TH applications should head for.

Almost all the users were amused at the beginning when the TH started to talk and addressed the user by his or her name. After a while though, a number of users seemed to be very distracted and not very interested in the game, some of them left rather soon. A reason for this might be the fact that the game started off with a quite long story, which did not require any interaction at all from the user. To encourage the users to continue listening until they got to the interaction part, this was explained to some of them.

When getting to the more interactive part of the game, almost all the users were eager to type in the actions they wanted to perform. The application was implemented in a way that it did not react to input until the full question was spoken. This lead to some confusion among some of the users. A difference between users was observed. The ones who seemed to be less familiar with computers waited patiently for the complete questions, while more experienced users were more eager. This indicates that some of the users were reading faster than the TH was speaking, i.e. that they might not have been listening that well to what the TH actually said. Though, when this question was asked, the users said they were both reading and listening. That could have been the case, but it might also have been caused by them trying to answer what they thought was the correct answer.

The fact that the same information was presented again if the user visited the same state of the game more than once, was pointed out as boring and annoying.

Considering the TH, most of the users liked the idea that the head actually talked to them, using their name. Some of them liked *John* and *Bernie* the most, because they were more realistic and professionally made, and some of them liked *Loris* better since, as they said, he was cooler and funnier.

Regarding the emotions in speech, the anger emotion seemed to be the easiest one to recognize. It was the only emotion that was pointed out. One person observed the changes in the voice but did not realize they were caused by attempts to express emotions.

Regarding the gestures expressed in the face, some users complained about the unnatural smile.

There was no obvious difference between boys and girls regarding their interest in the *Adventure Game*.

5.1.3 Conclusions

What was surprising, according to the expectations in advance, was the fact that so many users seemed to really enjoy the game. Since the users were excited about the interactivity, making the application more interactive will probably engage the users even more and might also get a larger number of people to become interested. A goal for the *Adventure Game* should be to get a larger percentage of the users to finish the whole game and not lose their interest. The users who actually finished the whole game were in general more enthusiastic about the game than others. This might be explained by the fact that these users got a real kick by managing to solve the riddle.

Users were really annoyed by the fact that they received the same information from the TH when a situation was repeated. This needs to be solved in some way. For instance by giving the user the opportunity to pass already visited areas more quickly, to minimize the information the second time or to give the information in some other way.

What kind of TH to be used in different kinds of applications is something that has to be considered. This evaluation shows that it is not always the most realistic looking head that is the best one to use.

Even though the users said they were both reading the text and listening to the head, it seemed like most of the users read rather than listened. If the aim is to have a hundred per cent attention to the TH, then how to present the information has to be taken into consideration. When the TH is not presenting any facts that are necessary for completing the task, the text might not be needed at all. Another solution might be to present the facts in some other way than plain text.

5.1.4 Outcome

After recommendations based on this evaluation, the *Adventure Game* was changed. All the conclusions above have not been taken into consideration in the new version, but the new feature added gives the users the possibility to move themselves to any stage in the game. This means that the users can start the game wherever they want, can skip the prologue if it is already known and can skip a number of stages in the game if these have already been visited.

The new version and the old version of the *Adventure Game* were compared at a trial with 25 students at the age of approximately 15. The project group did not perform the trial though. However, it was shown that the engagement from the students that tried

the new version of the game was much higher, i.e. when the duration between inputs from the user was shortened.

5.2 Applications

At the beginning of this project, the intended TH applications were a story teller and an information provider. The aim was to demonstrate the use of VHML as well as the DMT.

To be able to demonstrate the features of VHML appropriately, the applications have to include at least one TH that is marked up in VHML and that interacts with the user. During the research about THs at the beginning of the project, section 2.1, it was found that using THs in an application has a number of advantages. The ones that plead for using a TH in the story teller and the information provider are the following:

- Using THs in an application makes the human computer interaction become more like the conversation styles known from human-human communication.
- THs make an application more lively and appealing.
- THs make an application more compelling and easier to use, but only if they perform reasonable behavior, i.e. if the TH is implemented with respect to what people would expect from the same kind of creature in the real world regarding for example politeness, personality and emotion.
- THs can express nonverbal conversational and emotional signals.
- THs give personality to the application.
- People like being talked to.

The story teller was supposed to tell a story to the user, where the user could direct the story by answering questions posed by the TH. When starting to outline a story, the project group and the Interface group at Curtin questioned the whole idea of the application. One of the conclusions from the informal evaluation of the *Adventure Game*, section 5.1, was that more interactivity would engage the user even more. It was really hard to come up with a story that was interactive in an engaging way and therefore, the whole idea about the story teller was rethought. A new idea that came up was to instead develop a mystery application. By letting the user solve a mystery, the interactivity would increase significantly, since the user would be the one who poses the questions and therefore completely direct the conversation with the TH. The advantages with using a TH in a story teller or information provider application, plead for using THs in the mystery application as well. A mystery would also support the involvement of more than one TH, which is an advantage since the different THs can be allocated different personalities. This would make the THs more believable an engaging. Having more than one TH would also make it possible to demonstrate a wider spectra of VHML as well as the DMT, since the dialogues with different THs have to be combined.

The information provider was supposed to be an application providing information about THs, concerning MPEG-4, VHML and similar topics. When outlining the time schedules for the project the mystery felt more engaging to the project group than the information provider. It also turned out that the project group could not provide many facts concerning the application, since the project group did not have enough expertise regarding most of the topics. Therefore, the decision was taken to concentrate on the

mystery. The information provider has been developed to some extent, but since the project members have not played an active part in that, it will not be discussed further.

5.3 The Mystery at West Bay Hospital

The mystery application, *The Mystery at West Bay Hospital*, was developed during the project. Since this project is concerned with VHML and dialogue management, and does not include the actual creating of THs, the Interface group at Curtin developed the models for the application. The Interface group also implemented the underlying structure and connections. To get an overview of how the application works, both the GUI and the underlying structure will be described.

The original aim of the development of *The Mystery at West Bay Hospital* was to demonstrate the new VHML and the DMT. At the beginning of the project, the intent was that some employee at Curtin would implement VHML according to the new specification. Unfortunately, this has not been done and therefore the dialogue in the application has not been marked up in VHML.

While developing the application, the aim was still the original one, but when the application was finished, the aim of the evaluation changed according to the circumstances, section 6.3.

5.3.1 Background

One conclusion from the initial evaluation, section 5.1, was that the best model to use in a TH application is not always the most human-like one. Without further investigation concerning this, pictures of people in the Interface group were used as models for *The Mystery at West Bay Hospital*. The reason for this is that the Interface group at Curtin did not have access to as many TH models as were needed in the application and therefore new models had to be developed. Using pictures of people was the easiest and the least time consuming way to create completely new models.

The earlier evaluation indicated that text took user's attention from the TH. Further investigation is required regarding the best use of textual display with a TH. Since the goal for the user of the mystery application is to actually solve a mystery, the user might want to read earlier posed questions and corresponding answers more than once. Therefore, the text spoken by the TH is presented as plain text in addition to the spoken text.

To get some ideas about how a mystery can be designed, investigation of existing mystery applications on the web was made. A number of applications were found, with different stories and different design ideas. Some of them are described below.

- *Murder & Magic : Cluedo & Clue* (1997) is based on the classic board game *Cluedo* or *Clue*. The mystery application on the web concerns a murder that is to be solved by asking the six suspects questions. First, the user gets a summary of what has happened. By clicking different images of the characters and choosing among a number of predefined questions, the user gets answers from the suspects. When the user feels confident on who the murderer is, what the murder weapon is and in which room the murder was committed, the user makes a guess. If the answer is incorrect, it is possible to get hints on how to solve the mystery. The application includes drawn images of the suspects, murder weapons and rooms, but is otherwise totally text based. There are approximately seven questions to pose for each suspect.

- *The Detective's Chronicles Mystery Game* (2001) concerns a murder mystery to be solved by investigating the crime scene and interview the four suspects. At the beginning, the user is provided with a summary of what has happened. By clicking different images of the characters and choosing among a number of predefined questions the user gets answers from the suspects. The user can also visit the crime scene by clicking an image. When the user is confident on who the murderer is, the user makes a guess. The application is text based with some drawn images included. The number of possible questions to choose from for each suspect is three.
- At *Mysteries.com* (2001) there is one new murder mystery each day. The mystery starts off with an explanation of what has happened. After reading the story, the user can guess who the murderer is. The application is text based.
- At *MysteryNet.com* (2001) there is a murder mystery that includes an introducing story. The user can then guess who the murderer is and give an explanation why. The application is text based.
- *The Usual Suspects Vrml Mystery Game* (1997) is a 3D-based application. First, the user gets an introducing story. It is then possible to walk around in different scenes of the crime scene and try to figure out what has happened. This mystery is more like a game, i.e. if you are not careful you might, for example, get hit by a truck and die. The application is text based and includes drawn images of the suspects and the crime scene.

5.3.2 Design ideas

The design ideas of *The Mystery at West Bay Hospital* are similar to existing applications, in particular *Cluedo*, i.e. there are a number of suspects for the user to pose questions to, in order to solve the mystery. However, there are a number of differences between the two applications as well. The characters in *The Mystery at West Bay Hospital* are TH models, in contrast to the images used in all the applications mentioned above. Further, in *The Mystery at West Bay Hospital* the user will be able to pose any desirable question, instead of only choosing from predefined questions.

5.3.3 GUI

The Mystery at West Bay Hospital concerns a murder of one of the patients in a hospital. The full initial description about what has happened is included in Appendix G, this is also described to the user at the beginning of the mystery. The user plays the role of a private detective assisting a policeman to solve the mystery: Who murdered John Smith?

The policeman has some knowledge about what has happened and the user can pose questions to him, for example, concerning the crime scene or about the suspects. To help the user, there is a judge, who can give the user hints on how to find the murderer and give information on whether the user accuses the correct person or not. There are six suspects to whom the user can pose questions and the goal for the user is to find out which one of these suspects that has committed the murder. The GUI of the mystery application is shown in figure 25.

The application includes separate TH models for each person involved, i.e. the policeman, the judge and the six suspects.

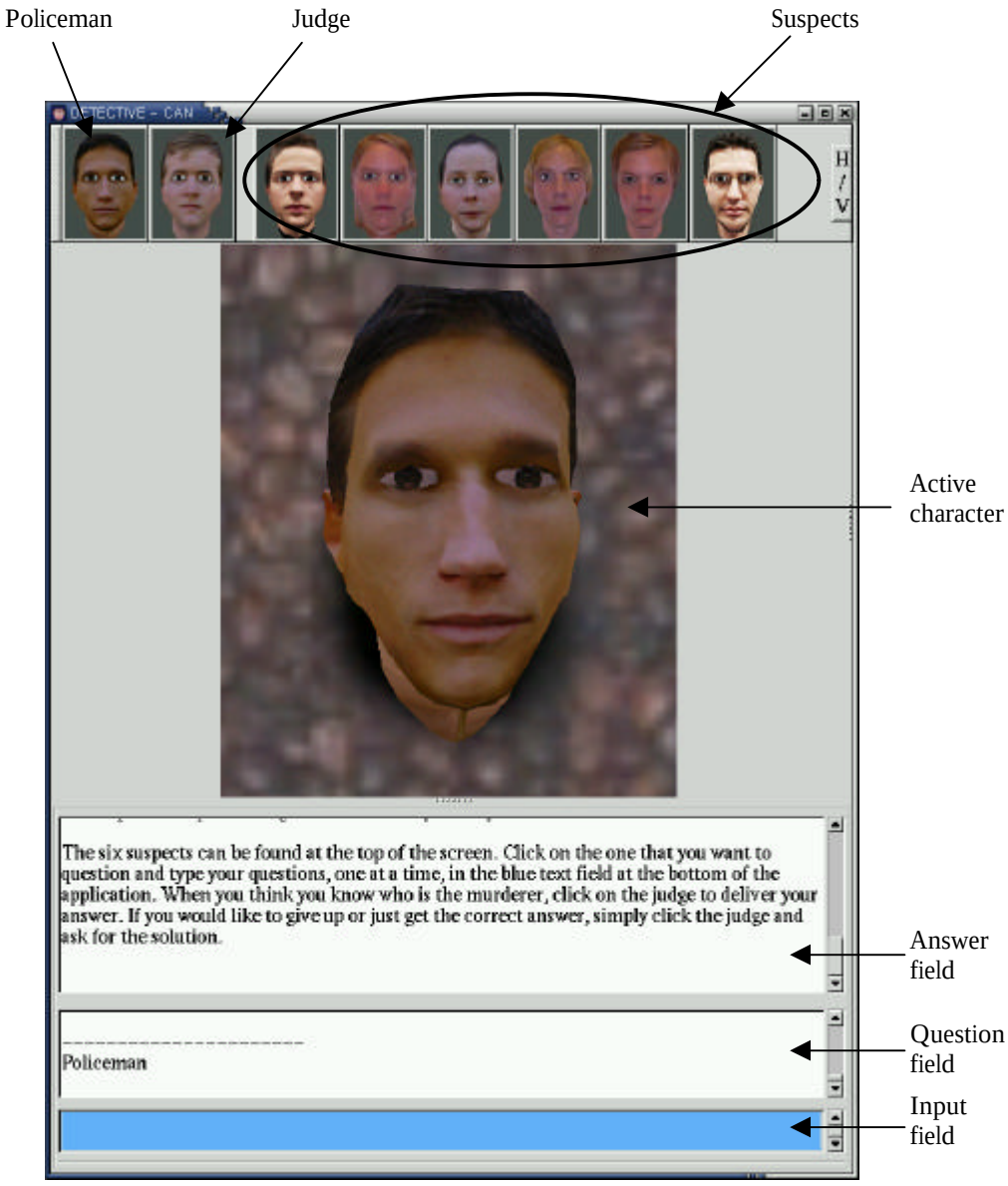

Policeman
Judge
Suspects
Active character
The six suspects can be found at the top of the screen. Click on the one that you want to question and type your questions, one at a time, in the blue text field at the bottom of the application. When you think you know who is the murderer, click on the judge to deliver your answer. If you would like to give up or just get the correct answer, simply click the judge and ask for the solution.
Answer field
Policeman
Question field
Input field

Figure 25. The Mystery at West Bay Hospital GUI.

At the top of the GUI, there are eight images of the characters involved in the mystery. Each image is connected to a tool tip that gives information about that particular character.

To pose a question to one of the characters, the user clicks the corresponding image, causing the character to appear at the center of the GUI, types in the question in the text field at the bottom of the GUI and then presses enter. The character responds to the question by speaking. The response is, as complementary to the spoken text, displayed in plain text below the image of the active character. In case the user wants to look back on previous questions and responses, it is possible to scroll up and down the answer and question fields.

To guess who the murderer is, the user clicks the image of the judge and types in the suggestion as above. Further, the judge can give the whole solution as well as some hints on how to solve the mystery if needed. The user gets three chances to guess who

the murderer is. It is also possible to get the correct solution before the application finishes.

5.3.4 Creating the dialogue

When developing an interactive TH application, the dialogue between the user and the TH is very important, section 2.8. The following steps were taken during the development of the dialogue in *The Mystery at West Bay Hospital*. The notation used is based on the DMTL DTD, Appendix D.

1. The dialogue was divided into nine different **<topic>** elements, where each corresponds to one of the eight characters that appear in the mystery and one is a general topic for the questions to which all characters should give the same answer. The reason for dividing the dialogue into topics is that the topics can be connected to a particular voice and to certain responses, depending on which character is active. Further, it gives a structure to the dialogue, which makes it easier to handle. A **<defaulttopic>** was created to take care of all input that are not covered by any other stimulus.
2. A dialogue network was created for each **<topic>**. These networks were only written on paper. The aim of the networks was to get an initial outline of each **<topic>** and to get similar structures in all **<topic>** elements.
   - The semantic of a number of conceivable questions were defined and connected to **<state>** elements.
   - The type of each **<state>** element was specified. The **<state>** elements that depend on earlier questions were defined as **linked** states and the ones that are independent as **entry** states. The **<state>** elements that do not need any user input to be triggered were defined as **active** states.
   - The connections between the **<state>** elements were specified, which correspond to the **<nextstate>** elements in the DMTL DTD. The approach to use **<nextstate>** instead of **<prestate>** was to suit the DM by Marriott at Curtin.
   - One **<stimulus>** and one **<response>** was specified for each **<state>**, just to know what kind of questions and responses each state would handle.
3. The dialogue networks were then implemented using the DMT, section 4.
   - If there were multiple **<state>** elements concerning the same topic, these were grouped to one **<subtopic>**.
   - To suit different variations of a question, the **<stimulus>** elements were generalized by implementing **<macro>** elements. In the fragment of the dialogue below, the values of the stimuli are specified as macros. For example, `KNOW` corresponds to all the possible ways of posing the semantic of the question "Do you know …" and `JOHN` corresponds to all the ways you can address the character John and so on.
   - One of the conclusions in the initial evaluation, section 5.1, was that different answers to the same question make the application less monotonous. Therefore, the number of **<response>** elements for each **<state>** was increased.
   - The characters were given personalities that influenced the **<response>** elements regarding expressions in speech.

- To be able to let the judge give different answers depending on how the mystery has been solved, boolean variables were set in the **<evaluate>** element in certain states. In this way the judge "knows" if these states have been visited. If none of the states that give proofs on who the murderer is have been visited, the judge knows that the user is just guessing and can give an appropriate answer. This also makes it possible to keep track of how many tries the user has made to guess who the murderer is.

5.3.5 A dialogue example

A fragment of the dialogue is shown below. The **<topic>** concerns the character Paul and will only be parsed when the DM looks for a matching stimulus if Paul is active, i.e. if the user has chosen to ask him questions by clicking his image in the top row of the GUI.

The example is describing two **<state>** elements in the **<subtopic>** concerning Paul's relation to John, the victim, more precisely concerning if Paul knows John.

The first **<state>** is an **entry** state, which means that the input can trigger the **<stimulus>** in this element at any time.

The **<stimulus>** is of the type **text**, since the input to the mystery application is text based. The value of the **<stimulus>** is two **<macro>** elements that have been combined to get the semantic intention "Do you know …?". The number of **<stimulus>** elements can be increased if needed.

The **<response>** has the response weight 0.7, since that is the default weight for responses in DMTL. When all the **<response>** elements have the same weight, which is the case in the example below, it is up to the DM to select the responses randomly. If the user input triggers the same state several times, the responses can then be different. The number of **<response>** elements can be increased. Since the responses include the XML entities, section 2.6.1, the content of the **<response>** element in this example is either plain text or empty with an attribute **statereference**. The **statereference** is a pointer to some other **<state>**, which means that the value of the **<response>** is the same for the **<state>** that is pointed to.

The **<nextstate>** elements define which **<state>** elements the dialogue can move into at the next step. The **entry** states can be moved into at any stage of the dialogue, therefore these do not have to be specified.

```
<topic name="Paul">
   …
   <subtopic name="relations">
      <subtopic name="John">
         <subtopic name="know">
            <state name="name" type="entry">
               <stimulus type="text">
                  KNOW(JOHN)
               </stimulus>
               <response weight="0.7">
                  Why should I know him, we are only sharing
                  room. That nerd was saying Good morning once a
                  day, but I never bother to answer. So I
                  can&amp;apos;t say I knew him very well.
               </response>
               <response weight="0.7">
                  I never knew that guy and I didn&amp;apos;t want
                  to either.
```

```
                </response>
                <nextstate name="Paul.relations.John.like.pron"/>
                <nextstate name="Paul.visitors.John.pron"/>
                …
            </state>
            <state name="pron" type="linked">
                <stimulus type="text">
                    KNOW(him)
                </stimulus>
                <response
                statereference="Paul.relations.John.know.name"/>
                <nextstate name="Paul.relations.John.like.pron"/>
                <nextstate name="Paul.visitors.John.pron"/>
                …
            </state>
        </subtopic>
        …
    </subtopic>
</topic>
```

5.3.6 Structure

The models of the different characters in the *The Mystery at West Bay Hospital* were developed as described in the work by Tschirren (2000). Firstly, two pictures were taken of the models, one from the front and one in profile. When building the models, the profile pictures were duplicated and used as both left and right side profile. Secondly, the pictures were mapped on a texture and attached to the model structure of a face. The models were then created as described in section 2.4.5.

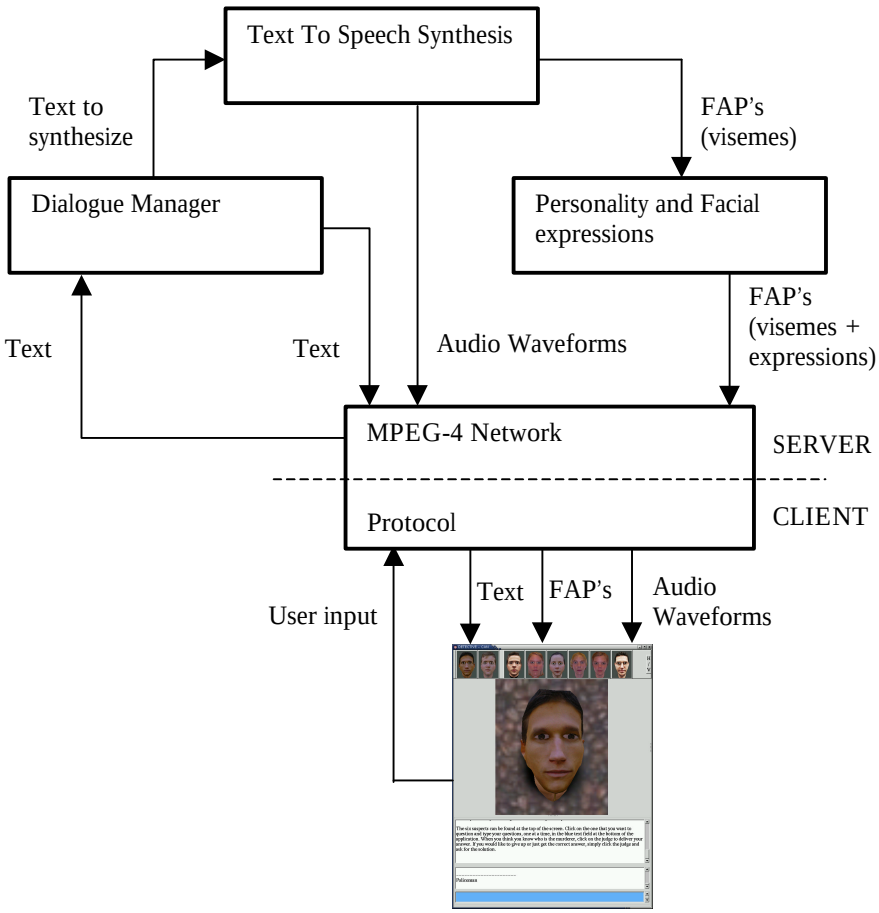


Figure 26. The underlying structure of The Mystery at West Bay Hospital.

The mystery is connected to a DM developed by Marriott at Curtin. The DM connects the input from the user to a certain stimulus, which then triggers the correct response. The entire structure of the application is shown in figure 26.

5.4 Discussion

There are several issues that can be further investigated and improved regarding both *The Mystery at West Bay Hospital* and the information provider, which finally was decided not to be a part of this project, but still will be developed within the Interface group.

- The dialogue in the mystery application grew rather large, reaching approximately 800 states. Further, the dialogue could be refined even more, probably for an infinite amount of time. The developer has to put an end to it somewhere, but the dialogue in the mystery application is not anywhere near being complete. The following can be considered:
    1. Include more states.
    2. Increase the number of responses in each state.
    3. Improve the stimuli.
- The initial evaluation pointed at the fact that the most realistic looking TH is not always the most appropriate one to use. Since this was not investigated further before the TH models in the mystery application were developed, there does still not exist a proof in any way that the realistic models are the best ones to use in this kind of application and this could be further investigated.
- During the initial evaluation a question arose whether or not to include text in TH applications. If text is not included, the important information has to be presented in some other way. How this could be made is not yet investigated.
- The users of the *Adventure Game* in the initial evaluation seemed to become very interested as soon as the TH started to address them with their typed in name. This could be a way to engage a user of the mystery application as well. This was not considered when the dialogue was created.
- In *The Mystery at West Bay Hospital* the crime scene is only described in words. Another possibility is to present a map of the crime scene, which would let the user investigate the crime scene by themselves.
- Since VHML is not yet implemented, the dialogue in *The Mystery at West Bay Hospital* has not been marked up with VHML. This is something that should be done as soon as the implementation has finished to evaluate VHML but also to make the application more engaging.
- The information provider, that was decided not to be a part of this project, has not yet been implemented to any greater extent. It is intended that the information provider will be developed in the same way as the mystery application. Therefore, the issues that arose during the development of the mystery application should be considered before the information provider is designed. This will prevent a lot of repetition of work with the information provider.

6 Evaluation

At the end of the project, evaluation of the work was performed for the three parts of the project. This since it is important to investigate whether or not the work is satisfactory, but also to give directions for future work within the areas.

6.1 VHML

The Interface group at Curtin is a part of InterFace and the development of VHML is important to the whole InterFace group. Since the members of this group are the first ones who will start using VHML once it is implemented, they were considered appropriate evaluators of the VHML Working Draft. VHML is described in section 1. Seven criteria have been the base when specifying VHML, i.e. completeness, simplicity, consistency, intuitivity, abstraction, usability and standardization, section 3.1. The aim of the evaluation was to find out whether or not the VHML Working Draft v. 0.3 (VHML v. 0.3, 2001) was considered fulfilling these criteria and thus to get feedback that can be of value for future work. The questionnaire that was sent to InterFace can be found in Appendix H.

6.1.1 Result

The respond from InterFace was not extremely satisfactory. The questionnaire was sent to fifteen partners with at least two members each, but only four of the questionnaires were returned. Though, these four gave good feedback and many hints for further improvements to VHML.

All contributors were asked to indicate their area/areas of expertise. This constituted the base for all comments given by the contributors. The areas that were covered in the returned questionnaires were: Image Synthesis, Speech Analysis, Speech Synthesis, Gestures, Emotions, Standards and Virtual Reality.

The questionnaire was separated into three major parts, the first covering the structure of the document, the second concerning the content of the VHML Working Draft v. 0.3 and the third for adding general comments that did not belong to any other section. The second part was divided into seven sub sections, one for each criterion that should be fulfilled.

The overall impression of the document structure was of satisfaction. Although, some opinions arose that should be considered for the next version of the VHML Working Draft.

- More code examples of complete VHML documents were requested to show the general structure of a valid document and at the same time demonstrate how useful and easy the language can be. This is especially a good way to make it easier for beginners to use VHML.
- The first section in the document, *Terminology and Design Concepts*, was experienced fairly complex and it was commented that it might scare the reader away before reading the rest of the document.
- A few concepts were unclear and hard to understand and should be explained more in detail. Among things mentioned were the **`variant`** attribute for **`<person>`** and also the **`<mark>`** element.

- It should be better explained what the relation is between **duration**, **wait**, having plain text between the start and end elements and having text after an empty element.
- Information about how all elements in the sub languages are related should be given more explicitly.
- Most contributors preferred a paper copy rather than an electronic version of the document. Though, with a printed copy, some features will get lost and therefore the specification should be available as an online document as well.

Some comments were given on the content of the language, which is also the most important issue for this evaluation. These were mainly concerned with the completeness of VHML and new features were proposed to fulfil this criterion.

- To cover all possible gestures and emotions, an extension mechanism for defining new gestures and emotions using low-level definitions such as FAPs, section 2.4.2, could be useful. This would probably mostly be used by the advanced users and thus increase the usability of the language.
- Hand movements should be added to the language.
- A way is needed to specify a skeleton and visual characteristics of the VH.

The contributors also found features decreasing the simplicity of VHML:

- There exist a **<mark>** element as well as a **mark** attribute for most of the other elements. If there is a reason for this duplication, it should be explained in the document and otherwise one of them should be removed.
- Instead of having one element for each direction, i.e. **<xxx-up>**, **<xxx-down>**, **<xxx-left>** and **<xxx-right>**, those can be combined to one element with direction in global space as attributes.

Regarding the abstraction level of the language, it was mostly found acceptable. However, one large obscurity arose:

- The usability of FAML was doubtful altogether. It was unclear if it was supposed to be on a very low abstraction level, as for FAPs, at a very high level, as for EML and GML, or anywhere in between those two levels. A suggestion was to express most FAML elements by defining low-level FAPs and merge some of the movements into GML on a higher level.

A number of valuable proposals, also concerning the content of the language, were gathered among the general comments:

- Perhaps it is too much freedom in the language. A validation mechanism could be implemented to prevent the possibility to define an animated face and body with different and inconsistent behavior, i.e. having a sad looking face with a happy looking body.
- EML elements include **duration** as an attribute. However, it should be possible to control the temporal characteristics of an emotion, i.e. how fast it appears and disappears. A good model for this may be to add the attributes **attack**, **sustain** and **decay**, where **attack** is the time for linear increase, **decay** the period of linear decrease and **sustain** the time in the middle where the top emotion level is sustained.

6.1.2 Discussion

The result from the evaluation turned out to be very valuable. Some of the opinions were very direct and easy to apply to VHML. Though, because of the time limit, some proposals have not been handled, but should still be considered in future development of the working draft. Moreover, some of the proposals concern already discarded issues, though these are also mentioned below.

A suggestion that came from three out of four of the responded contributors, was to include a code example of a complete VHML document. This is obviously a very important feature in order to make a specification easy to understand and consequently the language easy to use. To improve the document even further, some concepts should be explained more explicitly. It must be clear for all users how all elements and attributes should be used as well as the difference between using empty elements instead of start and end elements.

How elements from different sub languages are related to each other is demonstrated by links in the electronic version of the document. This information will get lost in a printed version, which seems to be the most common way of using a specification, and therefore all features in an online document should have a written corresponding explanation.

The first section of the document, *Terminology and Design Concepts*, is a leftover from the first version of the VHML Working Draft, which was given as a base for the work to be done in this project. Although, minor changes have been made to this section, it is still not clear enough and should therefore be rewritten from scratch.

Three of the proposals have already been discussed within the project. The discussion about why there is both a **<mark>** element as well as **mark** attributes can be found in section 3.3. The reason for having one element for each direction regarding the movement of the eyes and head is explained in section 3.6. A way to control the temporal characteristics of emotions and all other facial movements is an important improvement for making the VH as believable as possible. For doing so, three attributes should be added to all elements in EML, GML and FAML and eventually BAML, depending on how this sub language will be developed. These attributes can either be named after the model mentioned in section 2.3.3, **onset**, **apex** and **offset**, or the proposed model with the concepts **attack**, **sustain** and **decay**.

From people who are experts on gestures, it was proposed to add hand movements to VHML. This should either be a part of BAML or constitute a separate sub language, *Hand Animation Markup Language* (HAML). If this is to be added, detailed research has to be done in the hand gestures area.

The InterFace group is using MPEG-4 as a standard for the facial animation. Therefore, some suggestions arose concerning FAPs, which are the parameters used when animating a face according MPEG-4. One aim of the specification for VHML was to do it as general as possible, bearing in mind that it should not force the user to follow any animation standard in particular. Therefore, it is not appropriate, at this stage, to have a mechanism for defining new elements by using FAPs. However, if the aim of VHML will change and a decision is taken to use MPEG-4 as the standard for animating faces, this could be a very useful mechanism to add to the language. In that case, the FAML elements should also be lifted to a higher abstract level to differ them from the low-level FAPs.

A validation mechanism to prevent combining behaviors that make the VH less humanlike is a good suggestion. On the other hand, some actions are actually possible to combine even though they usually do not fit together. For example, a person might be sad and smile at the same time. To make the VH as believable as possible, detailed research has to be done in order to find out what combinations that should be prevented. The comment that it should not be possible to give the VH a sad looking face together with a happy looking body is not a problem though, since all emotion elements are inherited by both FAML for controlling the face as well as BAML for controlling the body.

6.1.3 Conclusions

Since VHML still is specified only as a working draft, many improvements can be done before it turns into a specification. Some of the issues that arose during the evaluation are already included in the working draft. An example of a complete VHML document including elements from all sub language has been constructed and placed at the end of the document and many of the concepts have also been described in more detail. This has resulted in VHML Working Draft v. 0.4, Appendix A, which is the present version to consult when using, but first implementing, VHML.

Some of the suggestions from the evaluation that were found useful will, because of the time limit, be considered as future work. The first section in the document should be rewritten to make the introduction simpler and more understandable. Hand movements should be added to VHML as well as other movements concerning the whole body. Moreover, a model for controlling the temporal characteristics for face movements should be added to all elements that affect the facial animation in some way. When implementing VHML, it should be considered which movements that do not work well together, and a validation mechanism for that should also be implemented.

If the aim of VHML remains the same, no consideration should be taken to the suggestions about defining new elements by using FAPs or expressing some FAML elements on a lower level. Though, if it will be decided to use MPEG-4 as a general base for all face movements in VHML, this can be of interest.

6.2 DMT

The DMT is a tool that the Interface group at Curtin needed for constructing dialogues when developing TH applications. The DMT is described in section 4. The tool has been developed in close cooperation with the Interface group and hence it is known that the DMT provides all current desirable functionality. Since there are no other potential users than the Interface group at the time, a formal evaluation has not been accomplished.

However, the DMT was used when the project group implemented the dialogue for *The Mystery at West Bay Hospital* and this section summarizes the thoughts that arose during that work.

The DMT GUI, described in section 4.3.2, was designed to fulfil a number of criteria, i.e. simplicity, consistency, intuitivity and usability. These criteria were considered during the informal evaluation. It should be pointed out though, that the members of the project group have both designed and evaluated the DMT. This may have affected the result.

6.2.1 Discussion

Overall, the DMT is very easy to use. The functionality is divided into a number of groups, i.e. different menus for different kinds of functions. Further, all functions that are provided in certain sections of the GUI are grouped together. This makes it easy to find the function you are looking for in the GUI.

The GUI is also consistent regarding names, positions of the elements, warnings and error messages.

The GUI feels intuitive, but this is not at all an objective assessment, since the project group has designed the DMT.

The usability is acceptable, although not perfect. There is a lack of keyboard shortcuts to the most used functions. The tool provides the possibility to tear off menus, which is kind of a shortcut. Further, it provides images in a toolbar, which as well are shortcuts. Though, keyboard shortcuts would be beneficial as well.

As soon as new information is added to a dialogue, parts of the GUI are repainted. But for some reason, the longer the DMT is used, the number of times that the GUI is repainted, for each time an update is made, increases. This leads to a large number of flashes, which is quite annoying.

It is not possible to use the scroll bars in the areas for including previous and next states, though it is possible to enter several different state references by using the arrow key on the keyboard.

When including a large number of macros, the list does not become scrollable. This means that it is not possible to see all included macros at once. This makes it difficult for the user to include new ones, since it is impossible to see if that specific macro already exists. It is also impossible to edit or delete the macros one cannot see. That makes it an unusable feature if there are many macros. The macros must be edited in an ordinary text editor instead.

The same thing happens when a dialogue contains many topics and subtopics. This is even worse since it means that these topics and subtopics cannot be used at all since one can not click on them in order to view their subtopics and states. Neither can they be edited nor deleted from the dialogue.

Further, when inserting the macros into the stimuli area, the list of macros is unsorted. It would be better if that list was sorted in alphabetical order to make it easier to find the macro to insert.

During the development of *The Mystery at West Bay Hospital* it was found that having parameters for the macros was very useful. This is a feature that should be included and it should be obvious which macros that require parameters and of which type these parameters should be.

Yet another feature for the macros that would be an advantage, is to be able to click a macro in the list to see which stimuli it contains. This makes it easier for a user who has included many macros and is uncertain what each macro contains.

The references typed in, i.e. in state reference, previous and next states areas, can become quite long since the fully qualified name has to be used. This is something that should be simplified in some way. A possible solution could be the scoping mechanism, section 4.4.

The dialogue in *The Mystery at West Bay Hospital* turned out to be fairly big. The number of states reached approximately 800. When inserting references in the state reference, previous and next states areas, the DMT performs checks on the fully qualified names. This was extremely useful, since human errors often occur regarding misspelling or when trying to make a reference to a state that does not exist.

When deleting or editing a topic, subtopic or state, the DMT deletes or changes all the references in the whole dialogue that point to this particular place. This is a very good feature, since the number of references might become very large and it also keeps the consistency in the dialogue since there does not exist any references to non-existing states.

One check that was missing though was if the macro element that the user types in really exists or not. It turned out to be a problem when the DM parses the DMTL document. This problem does not occur if the user at all times adds macros into the stimuli area using the list with provided macros. The problem only occurs when the user inserts macros by hand. Another problem is when macros are renamed or deleted. To simply remove all macro names that is not correct from the stimuli might not be good since that can cause weird phrases in the stimulus. But to let them remain in the stimuli causes inconsistency and problems for the DM. How to solve this problem has to be further investigated.

The DMT controls the structure of the dialogue. This made it very time efficient to use the DMT, since a minimum amount of typing was needed and since the DMT assures that the dialogue is a valid DMTL document at all times.

Though, when a DMTL document is created in an ordinary editor, it is possible to create a document with references to non-existing states. Currently, no checking is made when the DMTL document is opened in the DMT. So even if checks are made when new topics, subtopics and states are created inside the DMT, there can still be references to non-existing states within the dialogue. This is something that has to be checked in the future versions of the DMT.

The DMT gives a good overview of the dialogue concerning topics and subtopics. The states, though, are presented in a list with appropriate information. This presentation would have been even more useful if it was possible to view the elements in a network graph as well. In this way the connections between the elements would be easier to find.

The DMT only accepts a dialogue that is valid according to the DMTL DTD. Appropriate error messages and warnings make it impossible to implement an incorrect dialogue. This is of course a good feature, but if these error messages or warnings were not paid attention to, this sometimes caused loss of typed information. The reason for this is that if a warning is ignored, the information that is not correct is deleted to maintain the validity of the document.

In this version of the DMT, the copy, cut and paste functions are not implemented, section 4.7. This resulted in that an ordinary text editor was used when, for example, a next state element had to be inserted in a great number of states or when a dialogue was reorganised.

6.2.2 Conclusions

Even though there were some things within the DMT that could be improved, the overall impression of the DMT is that it is a very useful tool. It should be pointed out

that it is possible to create the dialogues without using any kind of tool. But by using the DMT the construction of the dialogues becomes much more time efficient since it makes it impossible to create an invalid DMTL document and because of the strict type control.

The most important improvements to consider are the use of macros and the listing of topics and subtopics. Maybe a complete rethought of how the macros are created, displayed and used is needed. The macros were introduced quite late in the implementation and therefore the implementation is not that good. In *The Mystery at West Bay Hospital* the macros were used quite frequently and if that is the case in most applications, the macros should be given priority in the further development of the DMT.

To not be able to display all topics and subtopics makes those features unusable if the dialogue grows too big, and therefore has to be solved.

Another improvement to the DMT with high priority is to remove the flashing of the GUI. The reason for the flashing has not been found, so that has to be investigated as well.

6.2.3 Talking Head workshop

In November 20th , the project group presented a paper, Appendix B, concerning the DMT at the Talking Head workshop that is a part of the *OZCHI* conference held in Fremantle 20-23 November, 2001 (Gustavsson, Strindlund & Wiknertz, 2001).

During the discussion after the presentation several issues arose, mostly concerning the DM that handles the output from the DMT. Since DMs are not a part of this project this will not be discussed further. However, some issues concerned the DMT as well.

Firstly, there was a question whether or not dynamical responses are possible to have inside the responses. If, for example, the stimulus is "How will the weather be in Perth today?" the response cannot be typed in advance since it will change from day to day. Instead, the response should be dynamic. A solution to this problem is to give the response "You can find information about the weather in Perth on the web site http://members.iinet.net.au/~jacob/weather.html." However, this is not a very nice solution and it does not actually give the user the information, it rather points to where this information can be found. A better way of doing this would be to have a command inside the responses, or even in the other field, that tells the DM to go to a certain web site and find the certain information and then present it to the user. This puts more pressure on the DM, but if the DM can handle this nicely the user will not notice the complexity behind it and will be satisfied with the answer.

Another question that was posed is whether or not there is a possibility for the DM to remember things about the user. For example, if the user previously has introduced him- or herself and then asks the question "What is my name?", the DM should have information about this and be able to answer the user correctly. Currently, this is not catered for in the DMT. However, if the dialogue includes a stimulus that matches the question above, it is possible to have mechanisms inside the responses that are connected to the user's name. One way of doing it is to use the sub language DMML in VHML. DMML is currently not specified, but the intent of the language is to cater for things like this. And when this is done it is up to the DM to handle it in a correct way.

6.3 The Mystery at West Bay Hospital

The original aim of the TH application developed within this project was to evaluate the new VHML and to demonstrate how to create dialogues using the DMT. This was, as discussed in section 5.3, changed since the new specification of VHML has not been implemented.

The mystery application has been developed, even though it was not possible to evaluate it as was thought originally. The Interface group at Curtin still requested an evaluation of the application. The evaluation was also performed in order to get directions for future development within its area. The aims of the evaluation was to find out:

- Whether the mystery was solvable or not.
- If the mystery was appreciated.
- If the dialogues within the mystery were correctly created.
- If all functionality in the application was sufficient.

The questionnaire was also constructed to give us information about whether the user's input was likely to be grammatically and structurally correct or not, if the users were used to solving mysteries and if they ever had used a TH application before.

It should be pointed out one more time though, that the mystery was not designed to be evaluated with these aims. If this had been the case, more effort would have been put in investigating how to create a correct and efficient dialogue. Since marking up the dialogues in VHML was the original objective of the evaluation, the content of the dialogues was not as important.

The questionnaire for the evaluation is attached as Appendix I. The evaluation was performed in cooperation with a PhD student at Curtin, Hanadi Haddad. Question numbers one to three are a part of her evaluation. Since the first one was quite interesting, the result to this is discussed even though it does not fulfil the aim of the evaluation of *The Mystery at West Bay Hospital*.

The evaluation was performed in a room at Curtin, with several computers and other people working. The participants were testing the application one at a time. They were first asked to read the front page of the questionnaire. Secondly, the policeman told the initial story, Appendix G, and thereafter the participant could start posing questions to the characters. The contributors were told to try to solve the mystery and that they could quit whenever they wanted.

6.3.1 Result

Seven people performed the evaluation, which included trying to solve the mystery and filling in the questionnaire, Appendix I. The result includes facts both from the questionnaire and the logged files from the application.

The questions from section 1, Personal and Background details, Appendix I, showed that the age of the participants was between 22 and 27 and it was one female and six males. Three of them had English as their first spoken language. One of them solved mysteries regularly and two of them had used TH applications before.

Two of the participants solved the mystery. The rest gave up with the reasons:

- Lack of time.
- Had not obtained any fresh information for some time.
- Lack of responses to questions.
- There was no leading in how to pose questions correctly.
- Not used to solve mysteries.
- Could not think of more questions to ask.

The contributors spent between 10 and 45 minutes in trying to solve the mystery.

All but one person guessed at least once on who the murderer was.

Four people asked the judge for hints, but the hints did not help them solve the mystery.

Everyone thought that the fact that the characters did not know the answers to many of the questions was annoying. Five of the contributors experienced that as slightly annoying, the other ones as annoying and very annoying.

Four of the participants found that all answers they got were relevant to the posed question. One did not answer and one of the remaining participants had the following example:

> Question: "Did you see anyone in John's room?"
> Response: The alibi for the concerned person.

Four of the participants found that it was possible to reword a question in order to get a satisfactory answer. Two of them said no and the last person did not try to do that.

All of the participants found *The Mystery at West Bay Hospital* from little to very much enjoying. Here are their comments:

- Lack of answers to questions that people are bound to ask and no real "leading" people towards questions that the characters *can* answer.
- Good hearing answers to questions I typed in, and to hear different sorts of responses (for example the Doctor was clinical and the roommate belligerent). Bit frustrating when you run out of questions.
- It's interesting to see talking heads able to pose relevant answers, as well as some realistic movement.
- Challenging and interesting (seeing how it has been set up).
- I think 30-40 minutes is not enough. Either one should have more time or there should be more examples/hints of how to ask questions, what kind of questions that can be asked. Apart from this I found the application interesting and fun.
- It's a fun game, need a little polishing to make it excellent.

The contributors found that the mystery was on an average to complicated complexity, which was mostly due to the lack of answers.

The following comments were collected as general comments:

- After a few minutes of actually trying to solve the mystery, I turned to trying to find questions that the characters could actually answer. After that, I just kind of gave up. The lack of direction is very frustrating and I had no idea that there were hints. Aside from that, the heads are nice, and work well in this sort of situations.
- The pop-up messages with each character's name were very useful as memory prompts – remind you who you're talking to or which one's the cleaner etc.
- I guessed the judge was next to the policeman, but that wasn't initially sure.
- Hit a few types or spelling mistakes.
- Different faces and voices important to enjoyment and story.
- Maybe should tell the player a bit about the game – so they can ask more relevant questions.
- I tended to read the responses rather than listen to them, which probably effected how I did remembering what they looked like.
- I probably didn't try to reword questions when I got the "I don't know" response because I assumed that the software was looking for keywords rather than the grammar of the statements (which is undoubtedly a bad thing to do).

Table 16 shows the information that was found from the logged files of each person performing the evaluation. The percentage is of the total number of questions minus the irrelevant ones, i.e. like "AAAAAAAAAAAAAAAARRRGGGH".

Person	Questions	Irrelevant questions	Correct answers	Wrong answers	Default answers	Time (m.s)
1	45	-	17 (37,8 %)	-	28 (62,2 %)	14.37
2	21	-	7 (33,3 %)	-	14 (66,7 %)	12.51
3	72	11	16 (26,2 %)	6 (9,8 %)	39 (63,9 %)	42.20
4	74	-	48 (64,9 %)	-	26 (35,1 %)	22.01
5 [1]	64	-	19 (29,7 %)	2 (3,1 %)	43 (67,2 %)	41.55
6 [2]	45	-	12 (26,7 %)	-	33 (73,3 %)	34.39
7	72	-	21 (29,2 %)	2 (3,8 %)	49 (68,1 %)	38.41

[1] the application crashed twice

[2] the person mixed up the names and called the victim for Paul for half the session

Table 16. Information from the logged files.

6.3.2 Discussion

The fact that contributor number 2, 5, 6, and 7 did not have English as their first spoken language might have affected the way they posed their questions. Since the DM by Marriott at Curtin did not check for keywords but for the grammar of the sentence these people may not have received answers on some of their questions because of writing errors.

Two of the participants marked that they had solved the mystery. According to the logged files they did not solve the mystery since they had not found enough evidence to convict the murderer, which means that they only guessed who the murderer was. This indicates that the question in the questionnaire was badly formulated.

Four people asked the judge for hints, but they did not found this helpful at all. Because the judge told them what to ask, they asked these questions but the characters just gave them the default answer that they did not know the answer to the question.

Except for one person, but he gave up anyway. The reason for this is either that the stimuli for some of the states were badly created or that the macros were not general enough.

Everyone marked the second best answer on the scale to the question if it was annoying that the characters did not know the answers to the questions. This was very surprising. According to the project group this fact was incredibly annoying. Exactly the same thing happened in the informal evaluation of the *Adventure Game*, section 5.1. The participants liked the application more than the project group did. The reason for this might depend on how used one is to TH applications. If it is the first time one sees a TH, maybe the expectations are lower. But this is not supported by the questionnaire though, when comparing the experience with TH applications and if the person enjoyed *The Mystery at West Bay Hospital*.

Four people found that they received relevant answers to all the questions that the TH did not answer with a default response. This means that the dialogue network does not contain that many direct errors.

Four participants also found that it was possible to reword the question in order to get the correct answer. This can both be seen as positive and negative. The positive thing is that the TH actually gave more correct answers because of this. The negative thing is that the stimuli and macros should be general enough to be able to handle all different sentences with the same intent, but obviously they are not. The stimuli are inserted in the dialogue, but making the macros and the rendering of the stimuli more general is up to the DM.

All the participants found that *The Mystery at West Bay Hospital* was of an average to complicated complexity. If the intent of this kind of application is to release it, the target group has to be decided. The participants in this evaluation were at the age of 22 to 27, which means that the application would not have been suitable for children.

6.3.3 Conclusions

Since there were only seven people in the evaluation, it is only possible to find trends in the result and discussion above. The result gives hints on what should be done with this application in the future, but it is not possible to draw any strong conclusions.

The DM was not perfect since the macros and stimuli did not seem to be general enough. One person pointed out that looking for keywords is the only way to go. After constructing, testing and evaluating *The Mystery at West Bay Hospital* the strong recommendation from the project group is to try the approach of looking for keywords instead of just pattern matching.

Before the evaluation, the project group did not find the dialogue anywhere near being complete and this is still the feeling. But even if this is a fact, the participants in the evaluation found that using THs in this kind of applications is very suitable and the THs were appreciated. The PhD student's questions concern how much the contributors remember of the THs. The answers to her questions have not been analysed in this project. However, one person pointed out among the general comments that he probably read more than he looked and listened to the THs. This might have affected the fact that he did not remember much about what the THs looked like. In the evaluation of the *Adventure Game*, section 5.1, it was also pointed out that investigation in how to present the information in addition to having a TH should be performed. This comment supports that issue even more.

The actual idea about having a story and a mystery that is to be solved seemed to engage the users. This is supported by the fact that several people put much effort in trying to solve the mystery, around 40 minutes.

7 Summary

The final outcome of the work done within this project is:

- A fourth version of the VHML Working Draft (VHML v. 0.4, 2001).
- A tool for making construction and maintenance of dialogues easier, DMT.
- A language for representing the dialogues, DMTL.
- A paper concerning the DMT and DMTL, (Gustavsson, Strindlund & Wiknertz, 2001).
- A TH application, *The Mystery at West Bay Hospital*.
- This Master thesis report.
- A presentation of the work that has been done.

The main aim of the project was to simplify the development of TH applications. To reach the aims of the project, research was made into many different areas; TH applications, facial animation, facial gestures, human speech, MPEG-4, XML and dialogue management.

The VHML Working Draft v. 0.1 (VHML v. 0.1, 2001) was examined in detail. The working draft was verified and validated and resulted in version 0.2 and 0.3. VHML Working Draft v. 0.3 (VHML v. 0.3, 2001) was evaluated by InterFace and the evaluation concluded with a fourth version of the VHML Working Draft (VHML v. 0.4, 2001).

The DMT was designed, implemented and tested by the project group. In order for DMT to represent a dialogue, an XML-based language, the DMTL, was specified. An informal evaluation of the DMT as well as the usage of the DMTL was made during the creation of the dialogue for *The Mystery at West Bay Hospital*.

The DMT and the DMTL was described in a paper presented at the Talking Head workshop at the *OZCHI* conference held in Fremantle, Australia at the 20th November 2001 (Gustavsson, Strindlund & Wiknertz, 2001).

The last aim in this project was to develop two separate interactive TH applications in order to show the advantages of using the DMT when constructing dialogues as well as the functionality of VHML. The two applications were supposed to be one story teller and one information provider concerning THs, MPEG-4 etc. The story telling application was changed to be a mystery instead, *The Mystery at West Bay Hospital*. This since the initial evaluation, section 5.1, showed that interactivity is an important feature and it was hard to find a story that was interactive enough, section 5.2. The information provider was decided not to be developed by the project group. This because of the time constraints in the project as well as the project group's lack of knowledge in some of the areas that the information provider should handle.

The initial purpose of *The Mystery at West Bay Hospital* was to demonstrate the new VHML and the DMT. This aim was changed during the project since VHML has not been implemented as planned. Though, when developing the mystery, the aim was the original one but the aim of the evaluation of the application changed according to the circumstances, section 6.3.

7.1 Future work

Because of the time limits of this project, some areas lack of investigation and there are many features that can be further improved. The development of all three parts of the project, VHML, DMT and *The Mystery of West Bay Hospital* will continue, even when this project is finished. In order to make it easier for those taking over the development, a number of issues have been gathered for each part as future work.

7.1.1 VHML

The future work in this area is based on the result of the evaluation, section 6.1, as well as already known issues that were not enough investigated because of the time constraints in the project, section 3.11.

Some of the sub languages have been given less attention than the other languages. To make VHML a more complete language, these sub languages have to be specified in detail. This involves research on the body movements for BAML, especially hand movements, which may lead to a new sub language, HAML. But also a research on what is needed for dialogue management and what additional elements that can be useful when controlling the text output, i.e. which subset of XHTML that should be included in VHML.

To increase the completeness of VHML, many of the sub languages can be expanded, though this can be done infinitely. However, this should not affect the simplicity of the language and must be done after careful research in respective area. The research should involve which movements that should be specified in FAML, which emotions that are useful for EML and which gestures GML should include.

At the moment, only certain parts of VHML are implemented. In the near future great effort will be given to implement the language according to the VHML Working Draft v. 0.4, which has been developed during this project. At that stage, many decisions have to be taken, for example, concerning the freedom of the language. These decisions are left to the programmer of the language to take.

7.1.2 DMT

There are several features to be improved in the DMT as well as new features that should be included. These have been found during the implementation, testing and informal evaluation of the DMT, sections 4.7 and 6.2. The major recommendations for future work are described here.

Firstly, the usage of the macros in the DMT has to be investigated. If it turns out that the macros are used as frequently as in *The Mystery at West Bay Hospital*, there must be investigated how they should be displayed and created.

Secondly, the display of the topics and subtopics in menus has to be rethought. It is not very useful to have the topics and subtopics listed in menus if all topics or subtopics cannot be seen. This has to be solved in some other way.

Further, it should be investigated what causes the GUI to flash. This does not affect the functionality of the DMT. However, as mentioned, it was found to be quite annoying, so it should be considered important when improvements are made to the DMT.

Moreover, the references to the states that are typed into the state reference, previous and next states areas are currently using fully qualified names. It was found out in the

informal evaluation that these names became quite long. A technique to avoid this can be to use the scoping mechanism and this has to be investigated.

At last, the functions for cutting, copying and pasting parts in a dialogue, for example, topics, subtopics or states as well as just plain text inside a state, were not implemented in this version of the DMT. This was a lack when the dialogue for *The Mystery at West Bay Hospital* was created and should therefore get high priority when improving the DMT.

7.1.3 The Mystery at West Bay Hospital

The future work that should be considered for *The Mystery at West Bay Hospital* are gathered from experience when developing and testing the application, section 5.4, as well as from the evaluation performed, section 6.3.

The actual idea about having a mystery that is to be solved seems to engage the users quite a lot. This is supported by the fact that several participants in the evaluation put so much effort in trying to solve the mystery, around 40 minutes, even though they could leave when they wanted.

There are still several issues that can be further investigated and improved regarding *The Mystery at West Bay Hospital*.

The dialogue in the mystery application grew rather large, reaching approximately 800 states. However, the dialogue could be refined even more, probably for an infinite amount of time. The developers have to put an end to it somewhere, but the dialogue in the mystery application is not anywhere near being complete. The following can be considered:

- Include more states.
- Increase the number of responses in each state.
- Improve the stimuli.

But even if it is a fact that the dialogue is not complete, the participants in the evaluation found that using THs in this kind of applications is very suitable and the THs were appreciated.

The initial evaluation, section 5.1, pointed at the fact that the most realistically looking TH is not always the most appropriate one to use. Since this was nsot investigated further before the TH models in the mystery application were developed, there does still not exist a proof in any way that the realistic models are the best ones to use in this kind of application and this could be further investigated. But the evaluation showed that the THs were appreciated, which support using more realistic THs in this kind of application.

During the initial evaluation, section 5.1, a question arose whether or not to include text in TH applications. If text is not included, the important information has to be presented in some other way. How this could be made is not yet investigated. One person in the evaluation pointed out that he thought he was reading more than looking and listening to the THs. This strengthens the suggestion that more investigation in this area is needed.

The users of the *Adventure Game* in the initial evaluation, section 5.1, seemed to become very interested as soon as the TH started to address them with their typed in name, which was also pointed out by the participants at the Talking Head workshop at the

OZCHI conference. This could be a way to engage a user of *The Mystery at West Bay Hospital* application as well. This was not considered when the dialogue was created.

In *The Mystery at West Bay Hospital* the crime scene is only described in words. Another possibility is to present a map of the crime scene, which would let the user investigate the crime scene by themselves.

Since VHML is not yet implemented, the dialogue in *The Mystery at West Bay Hospital* has not been marked up with VHML. This is something that should be done as soon as the implementation has finished in order to evaluate VHML, but also to make the application more engaging.

The DM was not perfect since the macros and stimuli did not seem to be general enough. One person in the evaluation pointed out that looking for keywords is the only way to go. After constructing, testing and evaluating *The Mystery at West Bay Hospital* the strong recommendation from the project group is to try the approach of looking for keywords instead of just pattern matching.

The overall opinion of *The Mystery at West Bay Hospital* is that the idea was very successful, but to get a more sophisticated application the dialogue has to be improved. The absolutely highest priority is to get the THs to answer a greater percentage of the posed questions, but all of the issues above should be investigated further.

Bibliography

Ananova (2000), *Ananova - www.ananova.com*. Available: http://www.ananova.com, [August 15, 2001].

André, E., Rist, T. & Müller, J. (1998a), 'Integrating Reactive and Scripted Behaviors in a Life-Like Presentation Agent'. In the proceedings of *The Second International Conference on Autonomous Agents (Agents '98)*, pp. 261-268, Minneapolis/St. Paul, USA.

André, E., Rist, T. & Müller, J. (1998b), 'Guiding the user through dynamically generated hypermedia presentations with a life-like character'. In the proceedings of *The 1998 International Conference on Intelligent User Interfaces*, pp. 21-28, San Francisco, USA.

Bates, J. (1994), 'The Role of Emotions in Believable Agents'. In *Communications of the ACM*, vol. 37, no. 7, pp. 122-125.

Beard, S. (1999), *FAQBot.* Honours thesis, Curtin University of Technology, Perth, Australia.

Beskow, J. (1997), 'Animation of talking agents'. In the proceedings of *AVSP'97 ESCA Workshop on Audio-Visual Speech Processing*, Rhodes, Greece.

Beskow, J., Elenius, K. & Mc Glashan, S. (1997), *The OLGA project: An animated talking agent in a dialogue system*. Available: http://www.speech.kth.se/~beskow/papers/fon97olga.html, [August 14, 2001].

Bickmore, T. W., Cook, L. K., Churchill, E. F. & Sullivan, J. W. (1998), 'Animated Autonomous Personal Representatives'. In the proceedings of *The Second International Conference on Autonomous Agents (Agents '98)*, pp. 8-15, Minneapolis/St. Paul, USA.

Binsted, K. (1998), 'Character Design for Soccer Commentary'. In the proceedings of *The RoboCup workshop, International Conferance on Multi-Agent Systems*, Paris, France.

Bosak, J. (1997), *XML, Java and the Future of the Web*. Available: http://webreview.com/1997/12_19/developers/12_19_97_4.shtml, [August 14, 2001].

Bosak, J. (1999), *The Birth of XML: A Personal Recollection*. Available: http://java.sun.com/xml/birth_of_xml.html, [August 14, 2001].

Bosak, J. & Bray, T. (1999), *XML and the Second-Generation Web*. Available: http://www.sciam.com/1999/0599issue/0599bosak.html, [August 14, 2001].

Bray, T. (1998), *Introduction to the Annotated XML Specification*. Available: http://www.xml.com/axml/testaxml.htm, [August 14, 2001].

Cahn, J. E. (1990), 'Generation of Affect in Synthesized Speech'. In *Journal of the American Voice I/O Society*, vol. 8, pp. 1-19.

Cardwell, A. (2001), *Review for Final Fantasy: The Spirits Within [2001]*. Available: http://us.imdb.com/Reviews/287/28795, [September 17, 2001].

Cassell, J. (2000), 'Embodied Conversation: Integrating Face and Gesture into Automatic Spoken Dialogue Systems'. In *Communications of the ACM*, vol. 43, no. 4, pp. 70-78.

Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Becket, T., Douville, B., Prevost, S. & Stone, M. (1994), 'Animated Conversation: Rule-Based Generation of Facial Expressions, Gesture and Spoken Intonation for Multiple Conversational Agents'. In the proceedings of *ACM SIGGRAPH '94*, Orlando, USA.

Cole, R., Massaro, D. W., de Villiers, J., Rundle, B., Shobaki, K., Wouters, J., Cohen, M. M., Beskow, J., Stone, P., Connors, P., Tarachow, A. & Solcher, D. (1999), 'New tools for interactive speech and language training: Using animated conversational agents in the classroom of profoundly deaf children'. In the proceedings of *ESCA/SOCRATES Workshop on Method and Tool innovations for Speech Science Education*, pp. 45-52, London, UK.

Duncan, S. (1974), *On the structure of speaker-auditor interaction during speaking turns.* Available: http://semlab2.sbs.sunysb.edu/Users/kryokai/duncan.html, [August 16, 2001].

Dutoit, T. (1997), *An Introduction to Text-to-Speech Synthesis.* Kluwer Acadesmic Publishers.

Ekman, P. (1979), 'About Brows: Emotional and Conversational Signals'. In *Human Ethology: Claims and Limits of a New Discipline*, ed. von Cranach, M., Fopps, K., Lepenies, W. et al., pp. 169-249, Cambridge University Press, New York.

Ekman, P. (1984), 'Expression and the nature of emotion'. In *Approaches to emotion*.

Ekman, P. & Friesen, W. (1975), *Unmasking the Face: A Guide to Recognizing Emotions from Facial Clues.* Prentice Hall, New Jersey.

GNOME Mailing Lists (2001), *The xml Archives.* Available: http://mail.gnome.org/archives/xml/2001-June/date.html, [August 15, 2001].

Gustavsson, C., Strindlund, L. & Wiknertz, E. (2001), 'Dialogue Management Tool'. In the proceedings of *The Talking Head Technology Workshop of OZCHI2001, The Annual Conference for the Computer-Human Interaction Special Interest Group (CHISIG) of the Ergonomics Society of Australia*, Fremantle, Australia.

Homer, A. (1999), *XML in IE5 Programmer's Reference.* Wrox Press Ltd., Birmingham.

Hougland, S. (2001), *Final Fantasy: The Spirits Within, Movie Review @ Hollywood.com*. Available: http://www.hollywood.com/movies/reviews/movie/471314, [September 17, 2001].

HumanMarkup.org (2001), *HumanMarkup org: Human Traits and Expression through XML*. Available: http://www.humanmarkup.org, [August 27, 2001].

Huynh, Q. H. (2000), *A Facial Animation Markup Language (FAML) for the Scripting of a Talking Head.* Honours Thesis, Curtin University of Technology, Perth, Australia.

InterFace (2001), *InterFace*. Available: http://www.ist-interface.org, [25 October, 2001].

Interface (2001), *Interface*. Available: http://www.interface.computing.edu.au, [September 7, 2001].

Ishizuka, L. (2001), *MPML Homepage Version 2.0e*. Available: http://www.miv.t.u-tokyo.ac.jp/MPML/en/2.0e, [August 27, 2001].

ISO/IEC (1998), 'Information Technology - Generic Coding of Audio Visual Objects - Part 2 : Visual, ISO/IEC 14496-2'. *Final Draft of International Standard*, October 1998. ISO/IEC JTC/SC29/WG11 Doc. N2502.

IST Programme (2000), 'B 3.4 Interface innovation in behavioural face and body synthesis'. In the proceedings of *Information Societies Technology (IST)*.

Knapp, M. (1980), *Essentials of Nonverbal Communication.* Harcourt College Publishers, Austin.

Koda, T. & Maes, P. (1996), 'Agents with Faces: The Effects of Personification of Agents'. In the proceedings of *HCI'96*, pp. 98-103, The British HCI Group, London, UK.

LifeFX (2001), *LifeFX: the Face of the Internet*. Available: http://www.lifefx.com, [August 16, 2001].

Lisetti, C. L. & Schiano, D. J. (2000), 'Automatic Facial Expression Interpretation: Where Human-Computer Interaction, Artificial Intelligence and Cognitive Science Intersect'. In *Facial Information Processing*, vol. 8, no. 1, pp. 185-235.

Lundeberg, M. & Beskow, J. (1999), 'Developing a 3D-agent for the August dialogue system'. In the proceedings of *AVSP'99*, Santa Cruz, USA.

Marriott, A. (to be published), 'A Java Based Mentor System'. In *Java in the Computer Science Curriculum*.

Marriott, A., Beard, S., Haddad, H., Pockaj, R., Stallo, J., Hyunh, Q. & Tschirren, B. (2000), 'The Face of the Future'. In *Journal of Research and Practice in Information technology*, vol. 32, no. 3, pp. 231-245.

Marriott, A., Pockaj, R. & Parker, C. (2001), 'A Virtual Salesperson'. In *Internet Commerce and Software Agents: Cases, Technologies and Opportunities*, eds Rahman, S. M. & Bignall, R. J., pp. 290-315, Idea Group Publishing.

Mauldin, M. L. (1994), 'Chatterbots, Tinymuds, And The Turing Test: Entering The Loebner Prize Competion'. In the proceedings of *AAAI-94*, AAAI Press, Seattle, USA.

Miller, P. W. (1981), *Non-verbal Communication.* National Education Association, Washington DC, USA.

MML (1999), *Music Markup Language*. Available: http://www.mmlxml.org, [November 19, 2001].

Moore, G. (2001), *Talking Heads: Facial Animation in The Getaway*. Available: http://www.gamasutra.com/features/20010418/moore_pfv.htm, [August 27, 2001].

Murder & Magic: Cluedo & Clue (1997), *Cluedofan.com (formerly Murder & Magic: Cluedo & Clue)*. Available: http://www.cluedofan.com, [February 1, 2002]

Murray, I. R. & Arnott, J. L. (1993), 'Toward the Simulation of Emotion in Synthetic Speech: A Review of the Literature on Human Vocal Emotion'. In *Journal of the Acoustical Society of America*, vol. 2, pp. 1097-1108.

Murray, I. R., Arnott, J. L. & Rohwer, E. A. (1996), 'Emotional stress in synthetic speech: Progress and future directions'. In *Speech Communication*, vol. 20, pp. 85-91.

Mysteries.com (2001), *Mysteries.com*. Available: http://www.mysteries.com, [September 20, 2001].

MysteryNet.com (2001), *The Online Mystery Network for everyone who enjoys a mystery*. Available: http://www.mysterynet.com/, [September 20, 2001].

Navarro, A., White, C. & Burman, L. (2000), *Mastering XML*. SYBEX Inc., Alameda, CA.

Pandzic, I. S. (2001 (to be published)), 'Life on the Web'. In *Software Focus Journal*.

Pandzic, I. S., Ostermann, J. & Millen, D. (1999), 'User evaluation: Synthetic talking faces for interactive services'. In *The Visual Computer Journal*, vol. 15, no. 7-8, pp. 330-340.

Pelachaud, C., Badler, N. I. & Steedman, M. (1991), 'Linguistic Issues in Facial Animation'. In *Computer Animation 1991*, pp. 15-30.

Pelachaud, C., Badler, N. I. & Steedman, M. (1994), 'Final Report to NSF of the Standards for Facial Animation Workshop'. 2001.

Pelachaud, C., Badler, N. I. & Steedman, M. (1996), 'Generating Facial Expressions for Speech'. In *Cognitive Science*, vol. 20, no. 1, pp. 1-46.

Pockaj, R. (1999), *FAP Specifications*. Available: http://www-dsp.com.dist.unige.it/~pok/RESEARCH, [August 2, 2001].

Poggi, I., Pelachaud, C. & de Rosis, F. (2000), 'Eye communication in a conversational 3D synthetic agent'. In *AI Communications*, no. Behavior planning for Life-Like Characters and Avatars.

Popick, J. (2001), *The Internet Movie Database [IMDb]*. Available: http://us.imdb.com/Reviews/287/28744, [September 17, 2001].

Reeves, B. & Nass, C. (1996), *The Media Equation*. Cambridge University Press.

Rist, T., André, E. & Müller, J. (1997), 'Adding animated presentation agents to the interface'. In the proceedings of *The 1997 International Conference on Intelligent User Interfaces*, pp. 79-86, Orlando, USA.

Sable (2001), *Sable 1.0*. Available: http://www.bell-labs.com/project/tts/sable.html, [September 3, 2001].

Sakaguchi, H. & Sakakibara, M. (2001), *FINAL FANTASY : THE SPIRITS WITHIN*. Available: http://www.finalfantasy.com, [September 17, 2001].

SAX 2.0 (2001), *The Simple API for XML*. Available: http://www.megginson.com/SAX/index.html, [August 10, 2001].

Scherer, K. L. (1996), 'Adding the Affective Dimension: a New York in Speech Analysis and Synthesis'. In the proceedings of *The International Conference on Speech and Language Processing (ICSLP 96)*, Philadelphia, USA.

Shepherdson, R. H. (2000), *The personality of a Talking Head*. Honours Thesis, Curtin University of Technology, Perth, Australia.

Stallo, J. (2000), *Simulating Emotional Speech for a Talking Head*. Honours Thesis, Curtin University of Technology, Perth, Australia.

Tekalp, M. & Ostermann, J. (1999), *Face and the 2-D Mesh Animation in MPEG-4.* Available: http://www.cselt.it/leonardo/icjfiles/mpeg-4_si/8-SNHC_visual_paper/8-SNHC_visual_paper.htm, [August, 15, 2001].

The Apache XML Project (2001), *The Apache XML Project*. Available: http://xml.apache.org/, [August, 15, 2001].

The Detective's Chronicles Mystery Game (2001), *The Detective's Chronicles Mystery Game*. Available: http://www.csd.uch.gr/~dtrip/index.html, [September 20, 2001].

The Usual Suspects Vrml Mystery Game (1997), *The Usual Suspects Vrml Mystery Game 1997*. Available: http://www.kahuna3d.com/games/UsualSuspects, [September 20, 2001].

The XML FAQ (2001), *The XML FAQ*. Available: http://www.ucc.ie/xml, [August 8, 2001].

Tschirren, B. (2000), *Realism and Believability in MPEG-4 Facial Models.* Honours Thesis, Curtin University of Technology, Perth, Australia.

VHML (2001), *VHML*. Available: http://www.vhml.org, [August 5, 2001].

VHML v. 0.1 (2001), *VHML Working Draft v. 0.1*. Available: http://www.vhml.org/document/VHML/2001/WD-VHML-20010925, [September 25, 2001].

VHML v. 0.3 (2001), *VHML Working Draft v. 0.3*. Available: http://www.vhml.org/document/VHML/2001/WD-VHML-20011021, [October 21, 2001].

VHML v. 0.4 (2001), *VHML Working Draft v. 0.4*. Available: http://www.vhml.org/document/VHML/2001/WD-VHML-20011123, [November 23, 2001].

VoiceXML (2000), *VoiceXML Forum.* Available: www.voicexml.org, [September 3, 2001].

W3C (1997), *Extensible Markup Language (XML).* Available: http://www.w3.org/XML, [August 16, 2001].

W3C (2001), *Speech Synthesis Markup Language Specification*. Available: http://www.w3.org/TR/speech-synthesis, [September 5, 2001].

Weizenbaum, J. (1976), *Computer Power and Human Reason.* W.H. Freeman and Company, New York.

Wierzbicka, A. (1992), 'Defining emotion concepts'. In *Cognitive Science*, vol. 16, pp. 539-581.

Wong, M. (2001), *Final Fantasy [2001]*. Available: http://www.moviem.com/reviews/F/finalfantasy.shtml, [September 17, 2001].

XML Standard API (2001), *XML Standard API: Package javax.xml.parsers.* Available: http://xml.apache.org/xerces2-j/javadocs/api/javax/xml/parsers/package-summary.html, [November 16, 2001].

XML White Papers (2001), *Introduction to XML*. Available: http://www.xml.org/xml/stpe_intro_to_xml.shtml, [August 8, 2001].

Glossary

API

Application Programming Interface. A series of functions that programs can use to make the operating system do their work.

BAML

Body Animation Markup Language. A sub language of VHML controlling the body movements for a VH.

DM

Dialogue Manager. An application handling dialogues between humans and computers.

DMML

Dialogue Manager Markup Language. A sub language of VHML supporting creation of dialogues with a VH.

DMT

Dialogue Management Tool. A tool that simplifies the construction and maintenance of a dialogue.

DMTL

Dialogue Management Tool Language. The language used when creating dialogues with the DMT.

DOM

Document Object Model. A standard tree-based API for XML and HTML documents.

DTD

Document Type Definition. A way to build up the grammar for an XML document that can be used to validate the document.

EML

Emotion Markup Language. A sub language of VHML controlling the emotions in speech, facial animation and body animation for a VH.

FAML

Facial Animation Markup Language. A sub language of VHML controlling the facial movements for a VH. But also the original Facial Animation Markup Language developed by Huynh.

FAP

Facial Animation Parameter. A parameter in a facial action to describe the deformation of a point from its neutral state.

FAPU

Facial Animation Parameter Unit. Spatial distances between major facial features on a face model in its neutral state.

FAQ

Frequently Asked Question. A commonly asked question and its answer.

FDP

Facial Definition Parameter. A set of parameters used for calibration of a face.

FP
Feature Point. A key-point in a human face.

GML
Gesture Markup Language. A sub language to VHML controlling the gestures of a VH.

HTML
HyperText Markup Language. A simple markup language used to create hypertext documents that are portable from one platform to another.

ISO
International Organization of Standardization. A worldwide federation of national standards bodies from some 140 countries, one from each country.

Meta-language
A language for describing other languages.

MPEG-4
A standard, defined by the *Moving Picture Experts Group*, for animating faces.

Namespace
A collection of names, identified by a URI reference, which are used in XML documents as element types and attribute names.

Qualified name
A name of an element in a tree hierarchy, defined as a concatenation of its local name and its preceding names back to the root.

Response
The output from the interactive application, depending on which stimulus that matches the input given by the user.

SAX
Simple API for XML. An event-based API for XML documents.

Scoping
A name is defined in the place where it is declared, but also within any other element that is declared within that element.

SGML
Standard Generalized Markup Language. A markup language controlling the presentation of information, but with more features than HTML.

SML
Speech Markup Language. A sub language of VHML controlling the speech of a VH. But also the original Speech Markup Language developed by Stallo.

SSML
Speech Synthesis Markup Language. An XML-based markup language for handling synthetic speech in web applications and other applications.

Stimulus
The match to the users input to an interactive application that is handled by the DM.

TH

Talking Head. A user interface consisting of an animated head that talks to the user.

TTS

Text To Speech. A synthesizer that translates text into speaking sound.

Validation

For an XML document to be valid, it has to follow the rules made up in the DTD.

VHML

Virtual Human Markup Language. A new markup language for controlling a VH, consisting of eight sub languages.

VH

Virtual Human. A character used in a user interface that interacts with the user.

Well formness

For an XML document to be well formed, the structure of it has to fulfil specific preconditions in order to be able to be interpreted and processed correctly in all applications.

VRML

Virtual Reality Modelling Language. A standard used for facial animation.

W3C

World Wide Web Consortium. An organization developing interoperable technologies for the Web.

XHTML

eXtensible HyperText Markup Language. A transition between HTML and XML. A subset of this is used as a sub language of VHML for controlling the presentation of text.

XML Schema

A way to build up the grammar for an XML document that can be used to validate the document.

XML

eXtensible Markup Language. A meta-language that is a small version of SGML and controls the presentation of information.

XSL

eXtensible Stylesheet Language. A powerful tool for transforming XML documents into other formats by transforming an XML document into a separate tree structure.

Index

A

a 62
A Bugs Life 25
abstraction 49
acoustic model 35
  articulation 35
  pitch 35
  timing 35
  voice quality 35
adjective 55
Adventure Game 83
affect display 29
afraid 42, 54
agree 42, 57
aim 17
Ananova 23
anchor 62
angry 42, 54
AntZ 25
apex duration 30, 97
API 39
application 22
  Adventure Game 83
  Ananova 23
  August 22
  Baldi 23
  FAQBot 23, 43, 65
  information provider 86
  interactive 19, 83, 86
  LifeFX 23
  Mentor System 65, 73, 75, 76
  Murder & Magic Cluedo & Clue 87
  Olga 22
  story teller 86
  The Detective's Chronicles Mystery Game 88
  The Mystery at West Bay Hospital 65, 87, 102
  The Usual Suspects Vrml Mystery Game 88
Application Programming Interface ..... *See* API
articulation 35
attribute 38
  name 38
  value 38
audio 43
August 22

B

background
  mystery 87
Baldi 23
BAML 45, 61
believability 25, 98
blink 44
Body Animation Markup Language ..... *See* BAML
body movement 61, 64
bored 42
break 43, 60

C

character data 37
cheek 30
chin 30
code 62
Colin 45
completeness 49
concentrate 57
conclusion
  DMT 100
  initial evaluation 85
  The Mystery at West Bay Hospital 105
  VHML 98
confused 42, 54
consistency 49, 76
conversational signal 28
criterion 49, 76, 95
  abstraction 49, 95, 96
  completeness 49, 95, 96
  consistency 49, 76, 95, 98
  DMT GUI 76, 98
  intiutivity 95
  intuitivity 49, 76, 98
  simplicity 49, 76, 95, 96, 98
  standardization 49, 95
  usability 49, 76, 95, 98
  VHML 49, 95
Curtin University of Technology 17

D

dazed 42, 54
default-emotion 54
defaulttopic 65, 67
  state 67
dialogue 65, 66, 90
  defaulttopic 66
  macros 66
  mystery 90
  topic 66
dialogue management 45
  Colin 45
  DM 46
  Eliza 45
  Julia 45
  response 45
  state 45
  stimulus 45
Dialogue Management Markup Language ..... *See* DMML
Dialogue Management Tool ..... *See* DMT
Dialogue Management Tool Language ..... *See* DMTL
Dialogue Manager ..... *See* DM
disagree 42, 57
discussion 62, 80
  DMT 80, 99
  initial evaluation 84
  The Mystery at West Bay Hospital 93, 104
  VHML 62, 97
disgusted 42, 54
DM 46, 92
  mystery 92
DMML 45, 62
DMT 17, 18, 19, 20, 65, 98
  conclusion 100
  criterion 76
  discussion 80, 99
  DOM 76
  edit 74
  evaluation 98
  export file 73
  fully qualified name 77
  future work 107
  GUI 73, 76, 79, 98
  help 75
  implementation 75
  import file 73
  Interface group 98
  Java 75
  JavaDoc 75
  macros 99, 101
  methodology 20
  nextstate 100
  open file 73
  options 75
  print file 73
  problem 77
  quit file 74
  requirements 73
  save file 73
  scoping 80
  shortcuts 99
  standard entity 78
  subtopic 100
  testing 79
  The Mystery at West Bay Hospital 98
  topic 100
  transform function 78
  user manual 80
  VHML 74
  view 75
DMTL 20, 65
  defaulttopic 65, 67
  dialogue 65, 66
  DOM 76
  DTD 65
  evaluate 65, 71
  example 71
  macro 65, 66
  macros 65, 66
  nextstate 65, 70
  other 65, 71
  prestate 65, 70
  response 65, 69, 78
  root element 66
  signal 65, 70
  state 65, 68
  stimulus 65, 69
  subtopic 65, 67
  topic 65, 67
  VHML 78
Document Object Model ..... *See* DOM
Document Type Definition ..... *See* DTD
DOM 39, 50, 76
  DMTL 76
  tree-based API 39
  VHML 50
double_blink 44

DTD 38, 49, 65
DMTL 65
VHML 49

E

ear 29
Eliza 45
embed 52, 63
emblem 28
EML 42, 54
adjective 55
afraid 42, 54
angry 42, 54
bored 42
confused 42, 54
dazed 42, 54
disgusted 42, 54
emotion 54
happy 42, 54
neutral 42, 54
noun 55
sad 42, 54
surprised 42, 54
wait 56
emotion 26, 32, 42, 54, 63
adjective 55
afraid 32, 54
angry 32, 54
blend 63
confused 54
dazed 54
disgusted 32, 54
happy 32, 54
neutral 54
noun 55
sad 32, 54
surprised 32, 54
universal 32
Emotion Markup Language ..... *See* EML
emotional emblem 28
emotional response 57
agree 42
disagree 42
emphasis 42
shrug 42
smile 42
emphasis 42, 43, 57
emphasise-syllable 43, 60
emphasize-syllable 43, 60
empty element 37
end element 37
evaluate 65, 71
evaluation 95, 98, 102
DMT 98
The Mystery at West Bay Hospital 102
VHML 95
event-based API 39
SAX 39
expressiveness 34
eXtensible HyperText Markup Language ..... *See* XHTML
eXtensible Markup Language ... *See* XML
eXtensible Stylesheet Language *See* XSL
eye 29
eye_down 44
eye_left 44
eye_right 44
eye_up 44
eye-blink 58
eyebrow 29
eyebrow_down 44
eyebrow_squeeze 44
eyebrow_up 44
eyebrow-down 58
eyebrow-up 58
eyes-down 58
eyes-left 58
eyes-right 58
eyes-up 58

F

facial animation 24, 31
A Bugs Life 25
AntZ 25
believability 25
Final Fantasy 25, 26
Tony de Peltrie 25
Toy Story 25
Facial Animation Coding System Markup Language ..... *See* FACSML
Facial Animation Markup Language ..... *See* FAML
Facial Animation Parameter ..... *See* FAP
Facial Animation Parameter Unit ..... *See* FAPU
Facial Definition Parameter ..... *See* FDP
facial expression 25, 28
affect display 29
conversational signal 28
emblem 28
emotional emblem 28
manipulator 29

punctuator ... 28
regulator ... 29
facial gesture ... 27
facial movement ... 58
facial part ... 29
cheek ... 30
chin ... 30
ear ... 29
eye ... 29
eyebrow ... 29
hair ... 30
head ... 30
mouth ... 29
nose ... 29
teeth ... 30
tongue ... 30
FACSML ... 55
FAML ... 18, 43, 58
blink ... 44
direction ... 58
double_blink ... 44
eye_down ... 44
eye_left ... 44
eye_right ... 44
eye_up ... 44
eye-blink ... 58
eyebrow_down ... 44
eyebrow_squeeze ... 44
eyebrow_up ... 44
eyebrow-down ... 58
eyebrow-up ... 58
eyes-down ... 58
eyes-left ... 58
eyes-right ... 58
eyes-up ... 58
facial movement ... 58
head_down ... 44
head_left ... 44
head_left_roll ... 44
head_right ... 44
head_right_roll ... 44
head_up ... 44
head-down ... 58
head-left ... 58
head-right ... 58
head-roll-left ... 58
head-roll-right ... 58
head-up ... 58
intensity ... 58
jaw-close ... 58
jaw-open ... 58
left_wink ... 44
look_down ... 44
look_left ... 44
look_right ... 44
look_up ... 44
look-down ... 58
look-left ... 58
look-right ... 58
look-up ... 58
repeat ... 59
right_wink ... 44
wait ... 56
which ... 59
wink ... 58
FAP ... 31, 96, 97
emotion ... 32
viseme ... 32
FAPU ... 33
FAQ ... 18, 69
FAQBot ... 23, 43, 65
FDP ... 34
Feature Point ... *See* FP
Final Fantasy ... 25, 26
FP ... 31
Frequently Asked Question ... *See* FAQ
fully qualified name ... 70, 71, 77, 80
DMT ... 99
scoping ... 80, 99
future work ... 107
DMT ... 107
The Mystery at West Bay Hospital ... 108
VHML ... 98, 107

G

gesture ... 57, 63
Gesture Markup Language ... *See* GML
GML ... 57
agree ... 42, 57
concentrate ... 57
disagree ... 42, 57
emotional response ... 57
emphasis ... 42, 57
gesture ... 57
repeat ... 57
shrug ... 42, 57
sigh ... 57
smile ... 42, 57
wait ... 56
Graphical User Interface ... *See* GUI
GUI ... 76, 79, 89
DMT ... 73, 76, 79, 98

mystery 89

H

hair 30
HAML 97
Hand Animation Markup Language ..... *See* HAML
hand movement 96, 97
happy 42, 54
head 30
head_down 44
head_left 44
head_left_roll 44
head_right 44
head_right_roll 44
head_up 44
head-down 58
head-left 58
head-right 58
head-roll-left 58
head-roll-right 58
head-up 58
HTML 36, 45
Human Markup Language ..... *See* HumanML
human speech 34
HumanML 55
HyperText Markup Language ..... *See* HTML

I

information provider 86
intelligibility 34
interactive 19, 83, 86
InterFace 17, 45, 55, 64, 95
Interface group 17, 41, 49, 52, 55, 66, 86
  DMT 98
  evaluation 102
International Organization of Standardization ..... *See* ISO
intuitivity 49, 76
ISO 31, 36

J

jaw-close 58
jaw-open 58
Julia 45

L

left_wink 44
LifeFX 23
limitation 19
Linköping University 17
logged files
  mystery 102
look_down 44
look_left 44
look_right 44
look_up 44
look-down 58
look-left 58
look-right 58
look-up 58

M

macro 65, 66
  stimulus 66
macros 65, 66
  macro 66
manipulator 29
mark 43, 52, 53, 95, 96
markup 37
markup element 37
Mentor System 65, 73, 75, 76
meta data 37
meta language 37
methodology 19
  demonstration 20
  DMT 20
  evaluation 20
  VHML 19
mouth 29
MPEG-4 ..... 31, 97
  FAP 31, 96, 97
  FAPU 33
  FDP 34
  FP 31
  neutral face 33
MPML 55
Multimodal Presentation Markup Language ..... *See* MPML
mystery 20, 65, 87
  background 87
  design 88
  dialogue 90
  DM 92
  GUI 89
  logged files 102

questionnaire 102
structure 92
Mystery at West Bay Hospital 20, 65, 87

N

namespaces 40, 50
neutral 42, 54
neutral face 31, 33
nextstate 65, 70, 100
fully qualified name 71
nonverbal behavior 27, 28
nose 29, 64
noun 55

O

offset duration 30, 97
Olga 22
onset duration 30, 97
other 65, 71
OZCHI 20, 101
workshop 20, 101

P

p 43, 52
xml:lang 53
paragraph 43, 52
xml:lang 53
pause 43
person 52, 63
variant 95
phoneme 43, 60
phoneme 30
pitch 43
pitch 35
pre 62
prestate 65, 70
fully qualified name 71
problem formulation 19
prosody 43, 60
prosody 34
punctuator 28

Q

qualified name 40
questionnaire 95, 102
mystery 102
VHML 95

R

regulator 29
response 65, 69, 78
statereference 69, 100
weight 69
response 45
result 102
The Mystery at West Bay Hospital 102
VHML 95
right_wink 44
root element 37, 42, 52, 66

S

s 43
Sable 43, 52
sad 42, 54
SAX 39, 76
event-based API 39
say_as 43
say-as 60
SCEE 55
schema 38, 49, 63
scoping 40, 80
fully qualified name 80, 99
sentence 43, 52
SGML 36
shrug 42, 57
sigh 57
signal 65, 70
significance 18
Simple API for XML ..... *See* SAX
simplicity 49, 76
smile 42, 57
SML 18, 43, 52, 60
audio 43
break 43, 60
emphasis 43
emphasise-syllable 43, 60
emphasize-syllable 43, 60
mark 43
p 43
paragraph 43
pause 43
phoneme 43, 60
pitch 43, 61
prosody 43, 60
rate 61
s 43
say_as 43

say-as.....60
sentence.....43
smooth.....60
speech.....60
target.....60
voice.....43, 60, 63
volume.....61
Sony Computer Entertainment Europe .....*See* SCEE
speaker model.....35
speech.....34, 60
acoustic model.....35
expressiveness.....34
intelligibility.....34
pitch.....35
prosody.....34
speaker model.....35
Speech Markup Language.....*See* SML
Speech Synthesis Markup Language .....*See* SSML
SSML.....18, 43, 50, 52, 60, 63
standard entity.....38, 78
Standard Generalized Markup Language.....*See* SGML
standardization.....49
start element.....37
state.....45, 65, 68
evaluate.....71
name.....68
nextstate.....70
other.....71
prestate.....70
response.....69
signal.....70
stimulus.....69
type.....68
state type.....68
active.....68
entry.....68, 70
linked.....68, 70
visitswitch.....68
statereference.....70, 100
fully qualified name.....70
stimulus.....65, 69
type.....69
stimulus.....45
stimulus type.....69
audio.....69
haptic.....69
text.....69
visual.....69
story teller.....86
stylesheet.....39
subtopic.....65, 67, 100
evaluate.....68
keywords.....67
name.....67
state.....67
subtopic.....67
surprised.....42, 54
synchronism.....30
apex duration.....30
offset duration.....30
onset duration.....30
Synthesis Speech Markup Language .....*See* SSML

T

Talking Head.....*See* TH
teeth.....30
TH.....17, 21
advantages.....21
Ananova.....23
application.....22, 83, 86
August.....22
Baldi.....23
drawbacks.....21
FAQBot.....23, 43
LifeFX.....23
Olga.....22
The Mystery at West Bay Hospital.....20, 65, 87, 102
conclusion.....105
discussion.....93, 104
DMT.....98
evaluation.....102
future work.....108
result.....102
timing.....35
tongue.....30
Tony de Peltrie.....25
top level elements.....52
topic.....65, 67, 100
name.....67
subtopic.....67
Toy Story.....25
transform.....50
tree-based API.....39
DOM.....39
TripleS Science Fair.....20, 83

U

universal emotions 32
usability 49, 76
user interface 21, 25

V, W

W3C 36
validation 38
well formness 38
VH 17
vhml 52
  xml:lang 53
VHML 17, 18, 19, 41, 49, 95
  BAML 45
  conclusion 98
  criterion 49, 95
  discussion 62, 97
  DMML 45
  DOM 50
  DTD 49
  embed 52, 63
  EML 42
  emotion 42, 54
  emotional response 57
  evaluation 95
  example 97
  FAML 43
  FAP 96, 97
  future work 98, 107
  hand movement 96, 97
  HTML 45
  levels 51
  mark 52, 53
  methodology 19
  nose 64
  p 52
  paragraph 52
  person 52, 63
  questionnaire 95
  result 95
  root element 52
  sentence 52
  SML 43
  speech 43
  sub language 41, 51
  temporal characteristics 96, 97
  top level elements 52
  transform 50
  validation mechanism 96, 98
  vhml 52
  XML Namespaces 50, 63
  XML Schema 49, 63
  XSL Stylesheet 50
VHML Working Draft v. 0.1 ..... 19, 41, 42, 49
VHML Working Draft v. 0.3 ..... 19, 49, 62, 95
VHML Working Draft v. 0.4 ..... 20, 49, 98
wink 58
Virtual Human ..... *See* VH
Virtual Human Markup Language ..... *See* VHML
Virtual Reality Modeling Language ..... *See* VRML
viseme 32
svoice 43, 60, 63
voice quality 35
VoiceXML 52
Working Draft v. 0.1 ..... 19, 41, 42, 49
Working Draft v. 0.3 ..... 19, 49, 62, 95
Working Draft v. 0.4 ..... 20, 49, 98
workshop 20
World Wide Web Consortium .*See* W3C
VRML 31

X

XHTML 62
  a 62
  anchor 62
  code 62
  href 62
  pre 62
  text output 62
XML 36
  attribute name 38
  attribute value 38
  character data 37
  declaration 37
  document 37
  DOM 39
  DTD 38
  empty element 37
  end element 37
  hierarchical order 38
  markup 37
  markup element 37
  meta data 37
  meta language 37
  namespaces 40, 50, 63
  parser 39

root element 37
SAX 39
schema 38, 49, 63
standard entity 38, 78
start element 37
stylesheet 39
validation 38
well formness 38
XML Namespaces 40
  XML Schema 49
  XSL Stylesheet 39, 50
XML Namespaces 40, 50, 63
  blending 40
  default 40
  qualified name 40
  scoping 40
XML Schema 38, 49, 63
XSL Stylesheet 39, 50

Appendix A

VHML Working Draft v. 0.4

Working Draft v0.4 **November 23th 2001**

This version:
http://www.vhml.org/documents/VHML/2001/WD-VHML-20011123/

Latest version:
http://www.vhml.org/documents/VHML/

Previous version:
http://www.vhml.org/documents/VHML/2001/WD-VHML-20011021/

Editors:

Camilla Gustavsson
Linda Strindlund
Emma Wiknertz
Simon Beard
Quoc Huynh
Andrew Marriott
John Stallo

Document maintainer:

vhml@vhml.org

Copyright © 2001 Curtin University of Technology, InterFace. All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.

Status of this document:

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the VHML website.

This is the 15th November 2001 Working Draft of the Virtual Human Markup Language Specification.

This working draft relies on the following existing languages:

Facial Animation Markup Language, developed by Huynh (2000).
Speech Markup Language, developed by Stallo (2000).
Speech Synthesis Markup Language,
http://www.w3.org/TR/speech-synthesis, developed by W3C.

The various sub languages of VHML use and extend these languages.

Abstract

This document describes the *Virtual Human Markup Language* (VHML). The language is designed to accommodate the various aspects of human computer interaction with regards to facial animation, text to speech production, body animation, dialogue manager interaction, emotional representation plus hyper and multi media information.

It uses existing standards and describes new languages to accommodate functionality that is not catered for.

The language is be XML/XSL based and consists of the following sub languages:

- EML Emotion Markup Language
- GML Gesture Markup Language
- SML Speech Markup Language (based on SSML)
- FAML Facial Animation Markup Language
- BAML Body Animation Markup Language
- XHTML eXtensible HyperText Markup Language
- DMML Dialogue Manager Markup Language (based on W3C Dialogue Manager or AIML)

Although general in nature, the intent of this language is to facilitate the natural and realistic interaction of a Talking Head or Virtual Human with a user via a web page or a standalone application. Specific intended use can be found in the deliverables of the InterFace project, http://www.ist-interface.org/.

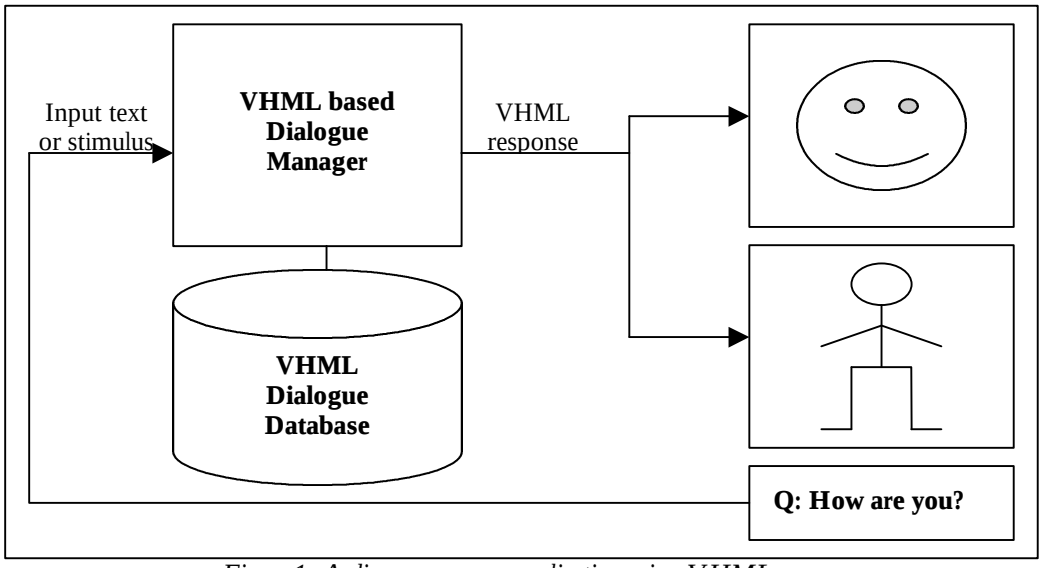


Figure 1. A diagram over an application using VHML.

Terminology and design concepts

The design and standardization process has adopted the approach of the Speech Synthesis Markup Requirements for Voice Markup Languages published December 23, 1999 by the W3C Voice Browser Working Group.

The following items were the key design criteria.

- *Consistency:* Provide predictable control of rendering output across platforms and across VHML implementations.
- *Generality:* Support rendering output for a wide range of applications with varied graphics capability and visual as well as speech content.
- *Internationalisation:* Enable visual and speech output in a large number of languages within or across documents.
- *Generation and Readability:* Support automatic generation and hand authoring of documents. The documents should be readable by humans.
- *Implementable:* The specification should be implementable with existing, generally available technology and the number of optional features should be minimal.

Rendering processes

A rendering system that supports the *Virtual Human Markup Language* (VHML) will be responsible for rendering a document as visual and spoken output and for using the information contained in the markup to render the document as intended by the author.

Document creation: A text document provided as input to the system may be produced automatically, by human authoring through a standard text editor, via a VHML specific editor or through a combination of these forms. VHML defines the form of the document.

Document processing: The following are the ten major processing steps undertaken by a VHML system to convert marked up text input into automatically generated output. The markup language is designed to be sufficiently rich so as to allow control over each of the steps described below, not necessarily in this order, so that the document author (human or machine) can control or direct the final rendered output of the Virtual Human.

4. **XML Parse:** An XML parser is used to extract the document tree and content from the incoming text document. The structure, elements and attributes obtained in this step influence each of the following steps.
5. **Culling of un-needed VHML elements:** For example, at this stage any elements that produce audio when the final rendering device or environment does not support audio may be removed. Similarly for other elements. It should be noted that since the timing synchronisation is based upon vocal production, the spoken text might need to be processed regardless of the output device's capabilities. This could be done via straight filtering or via XSLT.
6. **Structure analysis:** The structure of a document influences the way in which a document should be read. For example, there are common speaking and acting patterns associated with paragraphs.

- *Markup support:* Various elements defined in the VHML markup language explicitly indicate document structures that affect the visual and spoken output.

- *Non-markup behaviour:* In documents and parts of documents where these elements are not used, the VHML system is responsible for inferring the structure by automated analysis of the text, often using punctuation and other language-specific data.

7. **Text normalization:** All written languages have special constructs that require a conversion of the written form (orthographic form) into the spoken form. Text normalization is an automated process of the TTS system that performs this conversion. For example, for English, when "$200" appears in a document it may be spoken as "two hundred dollar". Similarly, "1/2" may be spoken as "half", "January second", "February first", "one of two" and so on. The same thing can appear for the body language. When somebody is saying "I caught a fish, this big" the person is supposed to show how big the fish is by using its hands.

- *Markup support:* The **<say-as>** element for speech, or **<do-as>** element for the body language, can be used in the input document to explicitly indicate the presence and type of these constructs and to resolve ambiguities. The set of constructs that can be marked includes dates, times, numbers, acronyms, duration and more. The set covers many of the common constructs that require special treatment across a wide number of languages but is not and cannot be a complete set. It has to be pointed out that there does not exist any body elements so far, but are seen as future work.

- *Non-markup behaviour:* For text content that is not marked with the **<say-as>** or **<do-as>** elements the TTS system is expected to make a reasonable effort to automatically locate and convert these constructs to a speakable and movable form. Because of inherent ambiguities (such as the "1/2" example above) and because of the wide range of possible constructs in any language, this process may introduce errors in the speech and body output and may cause different systems to render the same document differently.

8. **Text-to-phoneme conversion:** Once the system has determined the set of words to be spoken, it must convert those words to a string of phonemes. A *phoneme* is the basic unit of sound in a language. Each language (and sometimes each national or dialect variant of a language) has a specific phoneme set. For example, most US English dialects have around 45 phonemes. In many languages this conversion is ambiguous since the same written word may have many spoken forms. For example, in English, "read" may be spoken as [ri:d], "I will read the book" or [redd], "I have read the book".

Another issue is the handling of words with non-standard spellings or pronunciations. For example, an English TTS system will often have trouble determining how to speak some non-English-origin names, for example "Tlalpachicatl" which has a Mexican/Aztec origin.

- *Markup support:* The **<phoneme>** element allows a phonemic sequence to be provided for any word or word sequence. This provides the content creator with explicit control over pronunciations. The **<say-as>** element may also be used to indicate that text is a proper name that may allow a TTS system to apply special rules to determine a pronunciation.

- *Non-markup behaviour:* In the absence of a **<phoneme>** element the TTS system must apply automated capabilities to determine pronunciations. This is typically achieved by looking up words in a pronunciation dictionary and applying rules to determine other pronunciations. Most TTS systems are experts at performing text-to-phoneme conversions so most words of most documents can be handled automatically.

9. **Prosody analysis:** Prosody is the set of features of speech output that includes the pitch (also called intonation or melody), the timing (or rhythm), the pausing, the speaking rate, the emphasis on words and many other features. Producing human-like prosody is important for making speech sound natural and for correctly conveying the meaning of spoken language.

- *Markup support:* The **<emphasis>**, **<break>**, **<emphasize-syllable>** and **<prosody>** elements may all be used by document creators to guide the TTS system in generating appropriate prosodic features in the speech output.

- *Non-markup behaviour:* In the absence of these elements, TTS systems are experts (but not perfect) in automatically generating suitable prosody. This is achieved through analysis of the document structure, sentence syntax, and other information that can be inferred from the text input.

10. **Emotion analysis for speech, face and body:** Typically modify prosodic information before the *Digital Signal Process* (DSP). Some systems may wish to get access to data of their stage of the process.

11. **Waveform production:** The phonemes and prosodic information are used by the TTS system in the production of the audio waveform. There are many approaches to this processing step so there may be considerable platform-specific variation.

- *Markup support:* The TTS markup does not provide explicit controls over the generation of waveforms. The **<voice>** and **<person>** elements allow the document creator to request a particular voice or specific voice qualities, for example a young male voice. The **<embed>** element allows for insertion of recorded audio data into the output stream.

12. **Facial and body animation production:** Timing information will be used to synchronize the spoken text with facial gestures and expressions as well as with body movements and gestures.

13. **Rendering:** Rendering the multiple streams (Audio, Graphics, Hyper and Multi Media) onto the output device(s).

Document generation, applications and contexts

There are many classes of document creators that will produce marked up documents to be spoken and expressed by a VHML system. Not all document creators (including human and machine) have access to information that can be used in all of the elements or in each of the processing steps described in the previous section. The following are some of the common cases.

The document creator has no access to information to mark up the text. All processing steps in the VHML system must be performed fully automatically on plain text. The document requires only the root element to indicate the content is to be rendered.

When marked text is generated programmatically the creator may have specific knowledge of the structure and/or special text constructs in some parts of or the entire document. For example, an email reader can mark the location of the time and date of receipt of email. Such applications may use elements that affect structure, text normalization, prosody, possibly text-to-phoneme conversion, as well as facial or body gestures to gain the user's attention.

Some document creators make considerable effort to mark as many details of the document to ensure consistent speech quality across platforms and to more precisely specify output qualities. In these cases, the creator may use any or all of the available elements to tightly control the visual or speech output.

The most advanced document creators may skip the higher-level markup (emotions, facial and body animation tags) and produce low-level VHML markup for segments of documents or for entire documents.

It is important that any XML elements that are part of VHML use existing elements specified in existing (de facto) or developing standards (for example such as XHTML or SSML). This will aid in minimising learning curves for new developers as well as maximising opportunities for the emigration of legacy data.

The language structure

VHML uses a number of sub languages to facilitate the direction of a Virtual Human interacting with a user via a web page or a standalone application. These sub languages are:

- EML Emotion Markup Language
- GML Gesture Markup Language
- SML Speech Markup Language
- FAML Facial Animation Markup Language
- BAML Body Animation Markup Language
- XHTML eXtensible HyperText Markup Language (only a subset is used)
- DMML Dialogue Management Markup Language

VHML is divided into three levels, where only five elements constitute the top level. At the middle level are the two sub languages that control emotions and gestures, EML and GML. Their elements are inherited to three of the low level languages, SML, FAML and BAML. Apart from these three, there are two additional sub languages at the low level, DMML and XHTML. The structure of VHML is shown in figure 2. The dotted lines imply that the language on the lower level inherits the elements from the language on the upper level.

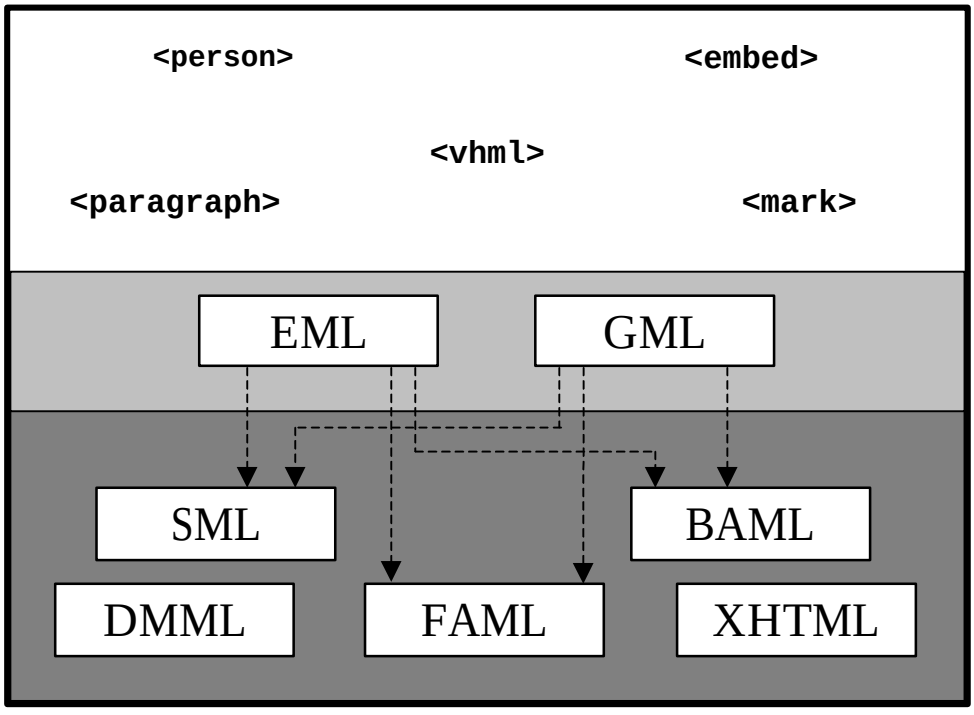


Figure 2. The structure of VHML.

In response to a user enquiry, the Virtual Human will have to react in a realistic and human way using appropriate words, voice, facial and body gestures. For example, a Virtual Human that has to give some bad news to the user may speak in a sad way, with a sorry face and a bowed body stance. In a similar way, a different message may be delivered with a happy voice, a smiley face and a lively body.

VHML is an XML based language. It uses a DTD in order to describe the rules of the structure of the language. The DTD for VHML is enclosed in Appendix A. As with XML elements, all VHML elements are case sensitive. Therefore all elements must appear in lower case and will otherwise cause fatal error. When creating a VHML document, the first line must contain an XML declaration followed by a DTD specification.

Example:
```
<?xml version="1.0">
<!DOCTYPE vhml SYSTEM "http://www.vhml.org/vhml.dtd">
…
```

For an example of a complete VHML document, it is recommended to look at section 0, Example of a VHML document.

Top level

The elements at the top level control the structure of the language as well as specify the speaker. An element used to embed foreign files is also placed on this level.

Top level elements

The following elements constitute the top level of VHML.

`<vhml>`

Description: Root element that encapsulates all other elements.

Attributes:

Name	Description	Values	Default
xml:lang	Indicates the language on the enclosing element.	a language code, following RFC1766	optional

Properties: Can only occur once.

Can contain **`<paragraph>`**, **`<mark>`** and **`<person>`** elements.

Example:

```
<vhml>
    …
</vhml>
```

`<person>`

Description: Specifies the speaker of the text, regarding gender, age and category as well as with which emotion it is supposed to speak and act in general. This emotion will constitute the default emotion for the rest of the element and is used whenever there is no other emotion specified.

Attributes:

Name	Description	Values	Default
age	Specifies the preferred age of the voice to speak the contained text.	integer	optional
category	Specifies the preferred age category of the voice to speak the contained text.	child teenager adult elder	optional
gender	Specifies the preferred gender of the voice to speak the contained text.	female male neutral	optional
name	Specifies a platform specific voice name to speak the contained text.	voice-name-list (a space separated list of names ordered from top preference down)	optional
variant	Specifies a preferred variant of another person to speak the contained text.	a character string that starts with the same string as the variant of the	optional

		person of which it should be a variant, then a colon and a name for that particular variant	
disposition	Specifies the emotion that should be used as default emotion for the contained text.	the name of any of the EML elements	optional

Properties: Can only occur directly under the **<vhml>** element.

Can contain **<paragraph>** and **<mark>** elements.

Note: If the attributes are not specified in the element, the values will be defined by the application itself and will therefore vary from application to application.

Even though the second person, as in the example below, is defined outside the first person element, the attributes to the first person are remembered. The variant of the person will then use the same attributes as the person it is a variant of, except if new attributes are specified for the second person. However, the variant will not look or sound exactly the same as the first person.

Example:
```
<vhml>
   <person age="12" gender="male" disposition="sad"
      variant="fred:1">
      …
   </person>
   <person variant="fred:2">
      …
   </person>
</vhml>
```

<paragraph> = <p>

Description: Element used to divide text into paragraphs. Both the whole word and the abbreviation can be used.

Attributes:

Name	Description	Value	Default
xml:lang	Indicates the language on the enclosing element.	a language code (following RFC1766)	optional
target	Specifies the destination for were the paragraph should be presented.	a character string	optional

Properties: Can only occur directly within a **<vhml>** element or a **<person>** element.

Can contain plain text as well as all other elements except itself, **<vhml>** and **<person>**.

Note: It is not possible to mix the abbreviation and the whole word for the same element, i.e. the start and end element must be in the same form.

The target attribute can be used for an application where something more than the Virtual Human and plain text should be presented. The value for target is dependent on the application.

Example:
```
<vhml>
   <paragraph>
      That was the weather for today.
   </paragraph>
   <p target="top">
      This is a summary of the weather forecast…
   </p>
   <p>
      Regarding the football game yesterday…
   </p>
</vhml>
```

<mark>

Description: Places a marker into the output stream for asynchronous notification. When the output of the VHML document reaches the mark, an event is issued that includes the name attribute. The platform defines the destination of the event. The mark element does not affect the speech or facial animation output process.

Attributes:

Name	Description	Value	Default
name	An identifier for the element.	a character string	required

Properties: Can occur in all non empty elements.

An empty element.

Note: Another way of placing a marker is by using the mark attribute that exists for all EML, GML, SML and FAML elements. The mark element can be used when a marker should be placed where there is no other element or at a global level in the document.

Example:
```
Go from<mark name="here"/>here, to<mark name="there"/>
there.
```

<embed>

Description: Gives the ability to embed foreign file types within a VHML document and for them to be processed appropriately.

Attributes:

Name	Description	Value	Default
type	Specifies the type of the embedded file.	audio mml	required
src	Gives the path to the embedded file.	a character string	required

Properties: Can occur in all non empty elements.

An empty element.

Example:
```
<embed type="mml" src="songs/Halleluja.mml"/>
```

Emotion Markup Language (EML)

The elements in EML will affect the emotion shown by the Virtual Human. These elements will affect the voice, face and body. All emotions will be inherited by SML, FAML and BAML.

EML default attributes

Each element has at least four attributes associated with it.

Name	Description	Value	Default
duration	Specifies the time span in seconds or milliseconds that the emotion will persist in the Virtual Human.	#s #ms (following CSSS)	required for empty elements and otherwise until closing element
intensity	Specifies the intensity of that particular emotion, either by a descriptive value or by a numeric value. Medium represents a numeric value equal fifty.	a numeric values (0-100) low medium high	medium
mark	Can be used to set an arbitrary mark at a given place in the text, so that an engine can report back to the calling application that it has reached the given location.	a character string that is an identifier for the tag	optional
wait	Represents a pause in seconds or milliseconds before continuing with other elements or plain text in the rest of the document.	#s #ms (following CSSS)	optional

Note: When both specifying a duration as well as using a closing element, the duration takes precedence over the closing element.

If the **`wait`** attribute is not specified the following text will start at the same time as the emotion. If wanting to start an emotion before continuing to speak, **`wait`** must be specified.

EML elements

The following elements constitute EML. All the universal emotions are included as well as neutral and two additional emotions.

`<afraid>`

Description: Generates a Virtual Human that looks afraid.

Facial animation. The eyebrows are raised and pulled together, the inner eyebrows are bent upward and the eyes are tense and alert.

Speech. The voice is not yet affected by this element.

Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly within the **`<paragraph>`** element.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<afraid intensity="40">
   Do I have to go to the dentist?
</afraid>
```

<angry>

Description: Generates a Virtual Human that looks and sounds angry.

Facial animation. The inner eyebrows are pulled downward and together, the eyes are wide open and the lips are pressed against each other or opened to expose the teeth.

Speech. The speech rate and pitch of stressed vowels are increased and the average pitch and pitch range are decreased.

Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly within the **<paragraph>** element.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<angry>
   You have to clean your room.
</angry>
```

<confused>

Description: Generates a Virtual Human that looks confused.

Facial animation. The eyebrows are bent upwards, the inner eyebrows are having great movement and the corners of the mouth are close together.

Speech. The voice is not yet affected by this element.

Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly within the **<paragraph>** element.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<confused duration="4s" intensity="high" wait="2s"/>
   Where did I put my keys?
```

<dazed>

Description: Generates a Virtual Human that looks dazed.

Facial animation. The eyebrows are slightly raised, the eyes opened somewhat wider than normal and the lips are slightly pulled down and outwards.

Speech. The voice is not yet affected by this element.

Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly within the **<paragraph>** element.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<dazed duration="10s"/>
   That was a tough sock you gave me.
```

<disgusted>

Description: Generates a Virtual Human that looks disgusted.

Facial animation. The eyebrows and eyelids are relaxed and the upper lid is raised and curled, often asymmetrically.

Speech. The voice is not yet affected by this element.

Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly within the **<paragraph>** element.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<disgusted intensity="80">
   I really hate chocolate cakes.
</disgusted>
```

<happy>

Description: Generates a Virtual Human that looks and sounds happy.

Facial animation. The eyebrows are relaxed, the mouth is open and the mouth corners pulled back towards the ears.

Speech. The speech rate, average pitch and pitch range are increased, so is the duration of the stressed vowels. The changes in pitch between phonemes are eliminated and the amount of pitch fall at the end of an utterance is reduced.

Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly within the **<paragraph>** element.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<happy duration="7s" wait="2000ms"/>
   It's my birthday today.
```

<neutral>

Description: Generates a Virtual Human that looks neutral.

Facial animation. All face muscles are relaxed, the eyelids are tangent to iris, lips are in contact, the mouth is closed and the line of the lips is horizontal.

Speech. The voice is not yet affected by this element.

Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly within the **<paragraph>** element.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<neutral wait="2s">
   I'm living in a red house.
</neutral>
```

<sad>

Description: Generates a Virtual Human that looks and sounds sad.

Facial animation. The inner eyebrows are bent upward, the eyes are slightly closed and the mouth is relaxed.

Speech. The speech rate, average pitch and pitch range are decreased. Abrupt changes in pitch between phonemes are eliminated and pauses are added after long words. The pitch for every word before a pause is lowered and all utterances are lowered at the end.

Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly within the **<paragraph>** element.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<sad intensity="low">
   I hurt my knee when I fell in the stairs.
</sad>
```

<surprised>

Description: Generates a Virtual Human that looks surprised.

Facial animation. The eyebrows are raised, the upper eyelids are wide open, the lower relaxed and the jaw is opened.

Speech. The voice is not yet affected by this element.

Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly within the **<paragraph>** element.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<surprised duration="2s" wait="500ms">
   I didn't expect to find that in my lasagne!
</surprised>
```

<default-emotion>

Description: The Virtual Human will get the emotion that is specified in the attribute **disposition** for **<person>**. If a person element does not exist, the emotion that is predefined for the application will be used.

Attributes: Default EML attributes.

Properties: Can only occur directly within the **<paragraph>** element.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<default-emotion>
   Now I'm talking in the same way as at the start.
</default-emotion>
```

Gesture Markup Language (GML)

The elements in GML will accommodate well-known human gestures. These will affect the voice, face and body of the Virtual Human. All gestures will be inherited by SML, FAML and BAML.

GML default attributes

Each element has at least four attributes associated with it.

Name	Description	Value	Default
duration	Specifies the time span in seconds or milliseconds that the emotion will persist in the Virtual Human.	#s #ms (following CSSS)	required for empty elements and otherwise until closing element
intensity	Specifies the intensity of that particular emotion, either by a descriptive value or by a numeric value. Medium represents a numeric value equal fifty.	a numeric value (0-100) low medium high	medium
mark	Can be used to set an arbitrary mark at a given place in the text, so that an engine can report back to the calling application that it has reached the given location.	a character string that is an identifier for the tag	optional
wait	Represents a pause in seconds or milliseconds before continuing with other elements or plain text in the rest of the document.	#s #ms (following CSSS)	optional

Note: When both specifying a duration as well as using a closing element, the duration takes precedence over the closing element.

If the **`wait`** attribute is not specified the following text will start at the same time as the gesture. If wanting to do a gesture before continuing to speak, **`wait`** must be specified.

GML elements

The following elements constitute GML.

`<agree>`

Description: Directs the Virtual Human to express "yes" or agreement by using gestures.

Facial animation. Animates a nod. It is broken into two sections, the head raise and then the head lower. Only the vertical angle of the head is altered during the element animation, the gaze is still focused forward.

Speech. The speech is not yet affected by this element.

Body. The body is not yet affected by this element.

Attributes: Default GML attributes.

Name	Description	Value	Default
repeat	Specifies how many times the action should occur.	integer	1

Properties: Can occur inside **<paragraph>**, EML, **<emphasis>**, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
That's certainly<agree duration="1000ms"/>right, Ollie.
```

<disagree>

Description: Directs the Virtual Human to express "no" or disagreement by using gestures.

Facial animation. Animates a shake of the head, which involves first moving to the left, then right and then returning to the central plane. The element only affects the horizontal displacement of the head and no other facial features are affected.

Speech. The speech is not yet affected by this element.

Body. The body is not yet affected by this element.

Attributes: Default GML attributes.

Name	Description	Value	Default
repeat	Specifies how many times the action should occur.	integer	1

Properties: Can occur inside **<paragraph>**, EML, **<emphasis>**, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<disagree intensity="20"/>
   I don't think you are right.
</disagree>
```

<concentrate>

Description: Directs a Virtual Human that has a concentrating look and sound.

Facial animation. The eyebrows are lowered and the eyes partly closed.

Speech. The speech is not yet affected by this element.

Body. The body is not yet affected by this element.

Attributes: Default GML attributes.

Properties: Can occur inside **<paragraph>**, EML, **<emphasis>**, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<concentrate wait="2s"/>
   Doing this is really a challenge.
</concentrate>
```

<emphasis>

Description: Emphasizes or accentuates words in the spoken text.

Facial animation. Animates a nod with the eyebrows lowering at the same rate.

Speech. The pitch and duration value are changed.

Body. The body is not yet affected by this element.

Attributes: Default GML attributes.

Name	Description	Value	Default
level	Specifies the strength of emphasis to be applied.	reduced none moderate strong	moderate

Properties: Can occur inside **<paragraph>**, EML, **<emphasis>**, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Note: When both **intensity** and **level** are specified, **level** takes precedence over **intensity**.

Example:
```
I will<emphasis level="strong">not</emphasis>buy this
record, it is scratched.
```

<sigh>

Description: Directs the Virtual Human to express a sigh.

Facial animation. The cheeks are puffed and also the eyebrows, head and mouth are affected.

Speech. The speech is not yet affected by this element.

Body. The body is not yet affected by this element.

Attributes: Default GML attributes.

Name	Description	Value	Default
repeat	Specifies how many times the action should occur.	integer	1

Properties: Can occur inside **<paragraph>**, EML, **<emphasis>**, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<sigh duration="2500ms" wait="2500ms"/>
We still have 2 km left on our walk.
```

<smile>

Description: Generates an expression of a smiling Virtual Human. It is generally used to start sentences and quite often when accentuating positive and cheerful words in a spoken text.

Facial animation. The mouth is widened and the corners pulled back towards the ears.

Speech. The speech is not yet affected by this element.

Body. The body is not yet affected by this element.

Attributes: Default GML attributes.

Properties: Can occur inside **<paragraph>**, EML, **<emphasis>**, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Note: A too large intensity value will produce a rather "cheesy" looking grin and can look disconcerting or phony.

Example:
```
<smile intensity="low"/>
   That was a beautiful dress you've got.
</smile>
```

<shrug>

Description: Mimics the facial and body expression "I don't know".

Facial animation. The head tilting back, the corners of the mouth pulled downward and the inner eyebrow tilted upwards and squeezed together.

Speech. The speech is not yet affected by this element.

Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Name	Description	Value	Default
repeat	Specifies how many times the action should occur.	integer	1

Properties: Can occur inside **<paragraph>**, EML, **<emphasis>**, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<shrug duration="5000" intensity="75"/>
I neither know nor care.
```

Speech Markup Language (SML)

The elements in SML affect the voice of the Virtual Human. The face and body will not be affected. The emotions will be inherited from EML and the gestures from GML.

SML default attributes

Each element has at least one attribute associated with it.

Name	Description	Value	Default
mark	Can be used to set an arbitrary mark at a given place in the text, so that an engine can report back to the calling application that it has reached the given location.	a character string that is an identifier for the tag	optional

SML elements

The following elements constitute SML.

`<break>`

Description: Controls the pausing or other prosodic boundaries between words. If the text is not marked up with the element, the speech synthesizer is expected to automatically determine a break based on the linguistic context, for example before starting a new sentence.

Attributes:

Name	Description	Value	Default
size	Specifies the duration of the break.	none small medium large	medium
smooth	Specifies if the last phoneme before the break has to be lengthened slightly.	yes no	yes
time	Specifies the duration of the break in seconds or milliseconds.	#s #ms (following CSSS)	optional

Properties: Can occur inside **`<paragraph>`**, EML, GML, FAML, **`<prosody>`** or **`<voice>`** elements.

An empty element.

Note: When both **`size`** and **`time`** are specified, **`time`** takes precedence over **`size`**.

Example: **`Well,<break size="large"/>I reckon this is a good idea.`**

<emphasize-syllable> = <emphasise-syllable>

Description: Emphasizes a syllable within a word. Both spellings of the tag can be used.

Attributes:

Name	Description	Value	Default
affect	Specifies how to emphasize the phoneme.	pitch duration both	pitch
level	Specifies the strength of the emphasis.	reduced none moderate strong	moderate
target	Specifies which phoneme in the text that should be emphasized.	a character string representing a phoneme symbol, using MPRA phoneme set	optional

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can only contain plain text.

Note: It is not possible to mix the two different spellings of the element, i.e. the start and end element must be in the same form.

Example:
```
I'm so<emphasize-syllable affect="duration"
level="strong" target="o">sorry.</emphasize-syllable>
```

<phoneme>

Description: Provides a phonetic pronunciation for the contained text.

Attributes:

Name	Description	Value	Default
alphabet	Specifies which phonetic alphabet that should be used.	ipa worldbet xsampa	optional
ph	Specifies the phoneme string.	a character string	required

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

The element may be empty, but it is recommended that the element contain human readable text.

Example:
```
I say tomato and you say<phoneme alphabet="ipa"
ph="t&#x252;m&#251;to&#x28A;>tomato</phoneme>.
```

`<prosody>`

Description: Controls the prosody of the contained text.

Attributes:

Name	Description	Value	Default
contour	Specifies the pitch contour for the contained text, with a percentage value of the period of the text (values outside the interval 0% to 100% are ignored) and a pitch, see the pitch attribute for values.	(interval, target), one or many pairs.	optional
duration	Specifies the desired time in seconds or milliseconds take to read the content of the element.	#s #ms (following CSSS)	optional
pitch	Specifies the baseline pitch for the contained text, either by a descriptive value or by a relative value representing the change to be done.	a numeric relative change (0-100) low medium high default	default
range	Specifies the pitch range for the contained text, either by a descriptive value or by a relative value representing the change to be done.	a numeric relative change (0-100) low medium high default	default
rate	Specifies the speaking rate for the contained text, either by a descriptive value or by a relative value representing the change to be done.	a numeric relative change (0-100) slow medium fast default	default
volume	Specifies the volume of the contained text, either by a descriptive value or by a relative value representing the change to be done.	a numeric relative change (0-100) silent soft medium loud default	default

Properties: Can occur inside **`<paragraph>`**, EML, GML, FAML, **`<prosody>`** or **`<voice>`** elements.

Can contain plain text as well as **`<embed>`** and **`<mark>`** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Notes: The default value of all the attributes is no change within the element compared to outside the element.

The **`duration`** attribute takes precedence over the **`rate`** attribute.

The **`contour`** attribute takes precedence over the **`pitch`** and **`range`** attributes.

Example:

```
<prosody contour="(0%,20)(10%,+30%)(40%,+10)>
   Good morning
</prosody>

<prosody rate="high" volume="high">
   I am talking very fast and very loud.
</prosody>
```

<say-as>

Description: Controls the pronunciation of the contained text.

Attributes:

Name	Description	Value	Default
type	Specifies the contained text construct. The format is a text type optionally followed by a colon and a format.	acronym number (ordinal, digits) date (dmy, mdy, ymd, ym, my, md, y, m, d) time (hms, hm, h) duration (hms, hm, ms, h, m, s) currency measure telephone name net (email, uri) address	required
sub	Specifies the pronunciation of the contained text.	a character string specifying the string that should be spoken.	optional

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can only contain plain text.

Example:

```
<say-as type="date:ymd">
   2001-09-06
</say-as>

<say-as sub="World Wide Consortium">
   W3C
</say-as>
```

<voice>

Description: Specifies the speaking voice of the contained text.

Attributes:

Name	Description	Value	Default
age	Specifies the preferred age of the voice to speak the contained text.	integer	optional
category	Specifies the preferred age category of the voice to speak the contained text.	child teenager adult elder	optional
gender	Specifies the preferred gender of the voice to speak the contained text.	female male neutral	optional
name	Specifies a platform specific voice name to speak the contained text.	voice-name-list (a space separated list of names ordered from top preference down)	optional

variant	Specifies a preferred variant of the other voice characteristics to speak the contained text.	integer	optional

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Notes: The **age** attribute takes precedence over the **category** attribute.

When there is not a voice available that exactly matches the attributes specified in the document, the voice selection algorithm may be platform specific.

Voice attributes are inherited down a tree structure.

The **variant** attribute does not work exactly the same as for **<person>**. For **<voice>** it is enough to give an integer as value and then a variant of the voice that encapsulates the element will appear.

Example:

```
<voice gender="male">
   Any male voice.
   <voice category="child">
      Any male child voice.
      <voice variant="2">
         This is another male child voice.
      </voice>
   </voice>
</voice>
```

The speech part of all elements belonging to EML is inherited to SML. To get the specification of the element, click on the tag and there is a link to the element described under the EML section.

<afraid> Inherited from EML.

<angry> Inherited from EML.

<confused> Inherited from EML.

<dazed> Inherited from EML.

<disgusted> Inherited from EML.

<happy> Inherited from EML.

<neutral> Inherited from EML.

<sad> Inherited from EML.

<surprised> Inherited from EML.

<default-emotion> Inherited from EML.

The speech part of all elements belonging to GML is inherited to SML. To get the specification of the element, click on the tag and there is a link to the element described under the GML section.

<agree>	Inherited from GML.
<disagree>	Inherited from GML.
<concentrate>	Inherited from GML.
<emphasis>	Inherited from GML.
<shrug>	Inherited from GML.
<sigh>	Inherited from GML.

Facial Animation Markup Language (FAML)

The elements in FAML affect the facial animation performed by the Virtual Human. These elements will only make changes to the face. The voice and body will not be affected.

The emotions will be inherited from EML and the gestures from GML.

FAML default attributes

Each element has at least four attributes associated with it.

Name	Description	Value	Default
duration	Specifies the time span in seconds or milliseconds that the emotion will persist in the Virtual Human.	#s #ms (following CSSS)	required for empty elements and otherwise until closing element
intensity	Specifies the intensity of that particular emotion, either by a descriptive value or by a numeric value. Medium represents a numeric value equal fifty.	a numeric value (0-100) low medium high	medium
mark	Can be used to set an arbitrary mark at a given place in the text, so that an engine can report back to the calling application that it has reached the given location.	a character string that is an identifier for the tag	optional
wait	Represents a pause in seconds or milliseconds before continuing with other elements or plain text in the rest of the document.	#s #ms (following CSSS)	optional

Note: When both specifying a duration as well as using a closing element, the duration takes precedence over the closing element.

If the **`wait`** attribute is not specified the following text will start at the same time as the movement. If wanting to do a movement before continuing to speak, **`wait`** must be specified.

FAML elements

The following elements constitute FAML.

All combinations of the directional elements allow the head to have full range of orientation. A combination of the **`<look-left>`** and **`<look-up>`** elements will enable to look at the top left in the animation sequence, whilst **`<look-right><look-down>`** will enable the head to look at the bottom right.

`<look-left>`

Description: Turns both the eyes and head to look left. The eyes and head move at the same rate.

Attributes: Default FAML attributes.

Properties: Can occur inside **`<paragraph>`**, EML, GML, FAML, **`<prosody>`** or **`<voice>`** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<look-left duration="1500ms" wait="1500ms"/>
Cheese to the left of me.
```

<look-right>

Description: Turns both the eyes and head to look right. The eyes and head move at the same rate.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<look-right>
   Cheese to the right of me.
</look-right>
```

<look-up>

Description: Turns both the eyes and head to look up. The eyes and head move at the same rate.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<look-up duration="5500ms" intensity="85" wait="2s"/>
Dear God, is there no escaping this smelly cheese?
```

<look-down>

Description: Turns both the eyes and head to look down. The eyes and head move at the same rate.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<look-down wait="2s">
Perhaps it is just my feet.
</look-down>
```

The eye directional elements allow four independent directions for eye movement. This entails movement in the vertical and horizontal planes. A combination of the **<eyes-left>** and **<eyes-up>** elements will enable to look at the top left in the animation sequence, whilst **<eyes-right><eyes-down>** will enable to look at the bottom right.

The eyes cannot be animated independently of each other.

<eyes-left>

Description: The eyes turn left, whilst the head remains in its position.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.
Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<eyes-left duration="1000ms" intensity="30" wait="1s">
There is the door, please use it.
```

<eyes-right>

Description: The eyes turn right, whilst the head remains in its position.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<eyes-right>
   A fly flew into my eye. Can you see it?
</eyes-right>
```

<eyes-up>

Description: The eyes turn upward, whilst the head remains in its position.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<eyes-up duration="4s" intensity="45"/>
You are just being foolish.
```

<eyes-down>

Description: The eyes turn downward, whilst the head remains in its position.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<eyes-down duration="3300ms" intensity="50"/>
Sorry for breaking your car.
```

The animation of the head movement can be broken down into three parts. The first affects the rotational angle of the head in the horizontal field, **<head-left>** and **<head-right>**. The second affects the elevation and depression of the head in the vertical field, **<head-up>** and **<head-down>**.The last affects the axial angle, **<head-roll-left>** and **<head-roll-right>**. The combination of these three factors allows full directional movement for the animation of the head of a Virtual Human.

<head-left>

Description: The head turns left, whilst the eyes remain in their position.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<head-left intensity="40">
   Do I have ice cream on my right cheek?
</head-left>
```

<head-right>

Description: The head turns right, whilst the eyes remain in their position.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<head-right duration="15s" intensity="40"/>
What about my left cheek?
```

<head-up>

Description: The head turns upward, whilst the eyes remain in their position.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<head-up duration="2s"/>
I'm a bit posh today.
```

<head-down>

Description: The head turns downward, whilst the eyes remain in their position.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<head-down wait="3s">
   Sorry, I'm ashamed of what I did.
</head-down>
```

<head-roll-left>

Description: Animates a roll of the head to the left in the axial plane. This is essential for adding realism to the Virtual Human and is often used in conjunction with other elements, such as agree and other head movements.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<head-roll-left duration="5s"/>
I have to stretch my neck.
```

<head-roll-right>

Description: Animates a roll of the head to the right in the axial plane. This is essential for adding realism to the Virtual Human and is often used in conjunction with other elements, such as agree and other head movements.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<head-roll-right duration="1500ms" wait="1s"/>
   Oh, what a cute dog you've got.
```

<eyebrow-up>

Description: Vertical movement upwards with the whole eyebrow. Eyebrow movements are especially used to accentuate words or phrases.

Attributes: Default FAML attributes.

Name	Description	Value	Default
which	Specifies which eyebrow to move.	both left right	both

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<eyebrow-up duration="3s" which="right"/>
I'm sceptical to what you say.
```

<eyebrow-down>

Description: Vertical movement downwards with the whole eyebrow. Eyebrow movements are especially used to accentuate words or phrases.

Attributes: Default FAML attributes.

Name	Description	Value	Default
which	Specifies which eyebrow to move.	both left right	both

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<eyebrow-down wait="2400ms">
   I'm really angry with you.
</eyebrow-down>
```

<eye-blink>

Description: Animates a blink with both eyes. Both the upper and lower eyelids are affected. The intensity value specifies how much of the eyes that should be closed.

Attributes: Default FAML attributes.

Name	Description	Value	Default
repeat	Specifies how many times the action should occur.	integer	1

Properties: Can occur inside **<paragraph>**, EML, GML, FAML , **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<eye-blink duration="40ms" repeat="2"/>
What a surprise!
```

<wink>

Description: Animates a wink of one eye. The wink is not just the blinking of one eye, but the head is affected as well as the outer part of the eyebrow and cheeks. The combination of these animated features add to the realism of the wink itself.

Attributes: Default FAML attributes.

Name	Description	Value	Default
which	Specifies which side to wink.	left right	left
repeat	Specifies how many times the action should occur.	integer	1

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
Nudge, nudge<wink duration="500ms" which="right"/>wink,
<wink duration="2000ms" which="right"/>wink.
```

<jaw-open>

Description: Opens the jaw on a Virtual Human.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
I'm really tired today.
<jaw-open duration="3s" wait="1s"/>
<jaw-close duration="2s"/>
```

<jaw-close>

Description: Closes the jaw on a Virtual Human.

Attributes: Default FAML attributes.

Properties: Can occur inside **<paragraph>**, EML, GML, FAML, **<prosody>** or **<voice>** elements.

Can contain plain text as well as **<embed>** and **<mark>** elements and all elements in GML, FAML, SML, BAML, DMML and XHTML.

Example:
```
<jaw-open duration="3s"/>
<jaw-close duration="2s"/>
I think I'm falling asleep.
```

The facial animation part of all elements belonging to EML is inherited to FAML. To get the specification of the element, click on the tag and there is a link to the element described under the EML section.

<afraid> Inherited from EML.

<angry> Inherited from EML.

<confused> Inherited from EML.

<dazed> Inherited from EML.

<disgusted> Inherited from EML.

<happy> Inherited from EML.

<neutral> Inherited from EML.

<sad> Inherited from EML.

<surprised> Inherited from EML.

<default-emotion> Inherited from EML.

The facial animation part of all elements belonging to GML is inherited to FAML. To get the specification of the element, click on the tag and there is a link to the element described under the GML section.

<agree> Inherited from GML.

<concentrate> Inherited from GML.

<disagree> Inherited from GML.

<emphasis> Inherited from GML.

<shrug> Inherited from GML.

<sigh> Inherited from GML.

Body Animation Markup Language (BAML)

The elements in BAML will affect the body animation performed by the Virtual Human. These elements will only make changes to the body. The voice and face will not be affected.

The emotions will be inherited from EML and the gestures from GML.

BAML elements

The following elements constitute BAML. No elements except them inherited from EML and GML have been included in the language.

The body animation part of all elements belonging to EML is inherited to BAML. To get the specification of the element, click on the tag and there is a link to the element described under the EML section.

<afraid>	Inherited from EML.
<angry>	Inherited from EML.
<confused>	Inherited from EML.
<dazed>	Inherited from EML.
<disgusted>	Inherited from EML.
<happy>	Inherited from EML.
<neutral>	Inherited from EML.
<sad>	Inherited from EML.
<surprised>	Inherited from EML.
<default-emotion>	Inherited from EML.

The body animation part of all elements belonging to GML is inherited to BAML. To get the specification of the element, click on the tag and there is a link to the element described under the GML section.

<agree>	Inherited from GML.
<concentrate>	Inherited from GML.
<disagree>	Inherited from GML.
<emphasis>	Inherited from GML.
<shrug>	Inherited from GML.
<sigh>	Inherited from GML.

eXtensible HyperText Markup Language (XHTML)

The elements in XHTML affect the output text from the application. Only a very limited subset of the actual XHTML is used in VHML.

XHTML default attributes

Each element has a number of attributes associated with it.

Name	Description	Value	Default
accesskey	Assigns an access key to the element.	a single character	optional
shape	Specifies the shape of a region.	default rect circle poly	optional
cords	Specifies the position and shape on the screen.	coordinates in percentage separated by commas	optional
tabindex	Specifies the position of the current element in the tabbing order for the current document.	0 – 32 767	optional
onfocus	Occurs when an element receives focus either by pointing device or by tapping navigation.	script data that can be the content of the script element and the value of intrinsic event attributes	optional
onblur	Occurs when an element loses focus either by pointing device or by tapping navigation.	script data that can be the content of the script element and the value of intrinsic event attributes	optional

XHTML elements

The following element constitutes the subset of XHTML that is used in VHML.

<anchor> = <a>

Description: Inserts an anchor in the output text.

Attributes:

Name	Description	Value	Default
charset	Specifies the character encoding of the resource designated by the link.	a space separated list of character encodings	optional
href	Specifies the location of a web resource, thus defining a link between the current element and the destination anchor.	a URI	optional
hreflang	Specifies the base language of the resource .	a language code, following	optional

		RFC1766	
name	Names the current anchor so that it may be the destination of another link.	a character string	optional
rel	Describes the relation from the current document to the anchor.	a space separated list of link types	optional
rev	Describe a reverse link from the anchor to the current document.	a space separated list of link types	optional
type	Gives a hint as to the content type of the content available at the link target address.	a content type, following RFC2045 and RFC2046	optional

Properties: Can occur inside all none empty elements.

Can only contain plain text.

Example:
```
Please look and find out on
<a href="http://www.vhml.org"> the VHML webpage</a>.
```

Dialogue Manager Markup Language (DMML)

Example of a VHML document

This is an example of a complete VHML document, using elements from all sub languages.

```
<?xml version="1.0">
<!DOCTYPE vhml SYSTEM "http://www.vhml.org/vhml.dtd">
<vhml>
   <person age="30" gender="male" disposition="sad">
      <p>
         <happy>
            I think that this is a great day.
            <smile duration="2s" wait="1s"/>
            <look-up>Look at the sky. There is
               <emphasislevel="strong">not a single
               </emphasis>cloud.
            </look-up>
            <agree duration="3500ms" repeat="4"/>
            The weather is perfect for a day at the beach.
         </happy>
         <angry intensity="60">
            But unfortunately my wife will say:
            <voice gender="female">
               This is<say-as type="date:md">0801</say-as>.
               The weather will probably be worse.
               Look at<a href=http://www.forecast.com/>the weather
                  webpage</a>to find out.
            </voice>
         </angry>
         <happy duration="5s" wait="1s"/>
            But I won't listen to her.
            Nudge, nudge <wink duration="450ms" which="right"/>wink
            <wink duration="550ms" which="right"/>wink.
      </p>
      <paragraph>
         <neutral>
            I'm very interested in music.
         </neutral>
         <default-emotion>
            This is a sad song. Listen to this.<break time="15s"/>
            <embed type="mml" src="music/sadLisa.mml"/>
            <eyes-down intensity="75">I usually start to cry<break
               smooth="no" time="1s"/>when I listen to it.
            </eyes-down>
            <prosody rate="slow" volume="soft">I think the lyrics
                  are<emphasis-syllable affect="both" target="ea">
                  really</emphasis-syllable>touching.
            </prosody>
         </default-emotion>
         <neutral mark="show_lyrics">
            If you look at the top left hand side <look-right
            duration="4s"/><look-up duration="4s" intensity="80"/>
            you can now read the lyrics.
         </neutral>
         <mark name="exit"/>
      </paragraph>
   </person>
</vhml>
```

References

Bradner, S. (1997), *Key words for use in RFCs to Indicate Requirement Levels*. Available: http://www.normos.org/ietf/rfc/rfc2119.txt [2001, September 12].

CSSS. Available: http://www.w3c.org/TR/RFC-C552/syndata.html [2001, October 14]

Faigin, G. (1990), *The Artist's Complete Guide to Facial Expression.* Watson-Guptill Publications, BPI Communications, Inc.

Fleming, B. & Dobbs, D. (1999), *Animating Facial Features & Expressions.* Charles River Media.

ftp://ftp.nordu.net/rfc/rfc1766.txt. Available: ftp://ftp.nordu.net/rfc/rfc1766.txt [2001, November 15].

Gustavsson, C., Strindlund, L. and Wiknertz, E. (2001), *Verification, validation and evaluation of the Virtual Human Markup Language (VHML).* Master Thesis, Linköping University, Linköping, Sweden.

Hyunh, Q. H. (2000), *A Facial Animation Markup Language (FAML) for the Scripting of a Talking Head.* Honours Thesis, Curtin University of Technology, Perth, Australia.

Java Speech Markup Language. Available: http://java.sun.com/products/java-media/speech/forDevelopers/JSML/index.html [2001, September 12].

Marriott, A. (2001), *InterFace.* Available: http://www.interface.computing.edu.au/ [2001, September 25].

Pelachaud, C. and Prevost, S. (1995), *Talking heads: Physical, linguistic and cognitive issue in facial animation.* Course Notes for Computer Graphics International '95.

RFC 1766. Available: http://www.nordu.net/ftp/rfc/rfc1766.txt

RFC 2045. Available: http://www.ietf.org/rfc/rfc2045.txt

RFC 2046. Available: http://www.ietf.org/rfc/rfc2046.txt

Sable V1.0. Available: http://www.research.att.com/~rws/Sable.v1_0.htm [2001, September 12].

Speech Synthesis Markup Language Specification. Available: http://www.w3.org/TR/speech-synthesis, [2001, September 13].

Speech Synthesis Markup Requirements for Voice Markup Languages. Available: http://www.w3.org/TR/voice-tts-reqs/, [2001, November 15].

Speech 2001. Available: http://www.microsoft.com/speech/, [2001, September 14].

Sproat, R. (1998), *The Proper Relation between SABLE and Aural Cascaded Style Sheets*. Available: http://www.bell-labs.com/project/tts/csssable.html, [2001, September 13].

Sproat, R., Hunt, A., Ostendorf, M., Taylor, P., Black, A., Lenzo, K. & Edgington M. (1998), *SABLE: A Standard for TTS Markup*. Available: http://www.research.att.com/~rws/SABPAP/sabpap.htm, [2001, September 13].

Stallo, J. (2000), *Simulating Emotional Speech for a Talking Head.* Honours Thesis, Curtin University of Technology, Perth, Australia.

TAGS AND ATTRIBUTES. Available: http://www.research.att.com/~rws/SABPAP/node2.htm, [2001, September 13].

Voice eXtensible Markup Language (VocieXML) version 1.0. Available: http://www.w3.org/TR/2000/NOTE-voicexml-20000505/, [2001, September 13].

VoiceXML Forum. Available: http://www.voicexml.org/, [2001, September 14].

Acknowledgements

Thanks to Ania Wojdel and Michele Cannella for their contribution with opinions about and proposed solutions to the structure of VHML.

We would also like to express gratitude to Igor Pandzic, Mario Gutierrez, Sumedha Kshirsagar and Jacques Toen, who are members of the European Union 5th Framework, for their comments during the evaluation of VHML.

Appendix B

Dialogue Management Tool

This is the paper presented by the project group November 20th 2001 at the Talking Head Technology Workshop of OZCHI2001, the Annual Conference for the Computer-Human Interaction Special Interest Group (CHISIG) of the Ergonomics Society of Australia in Fremantle, Australia.

Dialogue Management Tool

Camilla Gustavsson, Linda Strindlund, Emma Wiknertz
Linköping University, Sweden

Abstract

This paper describes a tool that can be used to simplify creating dialogues within, for example, an interactive *Talking Head* (TH) application or an ordinary question and answer file. What does the word *dialogue* within this area actually mean? Let us use the TH example; a dialogue occurs between the user and the TH when, for example, the user asks a question and the TH responds to that particular question. The answer given by the TH should be dependent on earlier questions and responses within that dialogue, i.e. which *state* the dialogue is in. A *Dialogue Manager* keeps track of the dialogue state and determines the responses to each question. But to be able to do this, the structure of the dialogue should be created in advance, i.e. all the different questions that the TH can answer should be defined, and these questions should be connected to the correct answer. To simplify the preparation of a dialogue, the *Dialogue Management Tool* has been developed. By using the tool the construction of the dialogues becomes easier, since it, among other things, prohibits incorrect references.

Keywords: Dialogue Management, Talking Head, FAQ, XML and Markup Language.

Introduction

In an interactive *Talking Head* (TH) application, there is a need for the TH to be able to converse with the user in some way. For example, a virtual salesperson has to be able to answer the user's questions about certain products. An information provider must answer questions about a certain domain. Furthermore, both have to actively ask questions or at least notify the user when it is unclear what the user really means.

Developing a dialogue includes creating *stimuli* and *responses*. When the user input matches a stimulus this should trigger the correct response. Depending on the stimulus the dialogue should traverse into different *states*. This is a well-known trick to make an application seem more intelligent. By handling this, the application will know the context of the dialogue and will therefore be able to respond correctly. The trick has been used by, for example, *Julia* and *Colin*, who are two chatterbots developed by Mauldin (1994). They seem somewhat intelligent to the user even though the structure of their knowledge is an ordinary network with a number of states.

Managing the dialogue is a very important issue in order to create an interesting and interactive TH application. By using network structures for the dialogue it is possible to create a more intelligent conversation since it gives the possibility to keep track of the conversation's state. Since the dialogues might become very large and complex, it can take a great amount of time to construct correct network structures. The aim of the *Dialogue Management Tool* (DMT) is to simplify the construction and maintenance of the dialogue.

Representation of a dialogue

The TH in the following dialogue between a TH and Anna uses the same trick as *Julia* and *Colin*, i.e. moves the dialogue into different states depending on Anna's input:

TH says, "How are you?" *to Anna.*
Anna says, "Not so good." *to TH.*
TH says, "Why is that?" *to Anna.*
Anna says, "I have a terrible headache." *to TH.*
TH says, "Have you taken aspirin?" *to Anna.*
Anna says, "I have to go. Goodbye!" *to TH.*

Figure 1 represents a fragment of the rules used by the TH in the discussion. In the diagram, S represents the stimulus, written in a regular expression, and R represents the response.

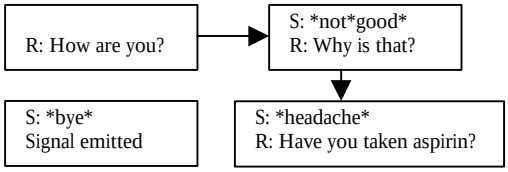


Figure 1. A diagram of the greeting example.

The first question is an active prompt from the TH and does not have to be triggered by a stimulus. Anna's answer, "Not so good." is a stimulus that moves the dialogue to a different state. In this new state the TH knows that Anna is not feeling good. The TH then asks: "Why is that?", which is a response that only can take place because of the fact that the TH "remembers" the previous questions and answers. Anna's answer about the headache is yet another stimulus that moves the dialogue into a new state and a responding question is posed.

Anna's end phrase moves the dialogue into a final state, which also is an entry state and therefore can be entered at any time during the dialogue.

This short example points out the importance of dividing the dialogue into different states. The question "Why is that?" can not be posed without a known context, since it would not have a meaning if the context is missing. Furthermore, to pose the question "Have you taken aspirin?" the TH has to know that Anna suffers from a headache. It is also important to point out that the TH can keep track of a whole sequence of stimuli and responses. This means that the TH can produce a response that relates to a discussion that appeared earlier in the conversation.

The user input might contain grammatically incorrect stimuli, but it should still trigger a response. Using pattern matching for the stimulus input solves this. Furthermore, a certain response might be considered the "correct" one for more than one stimulus. In the previous example, the stimuli "Not so good." should trigger the same response as for example "I'm not feeling very well today." and hence give the same answer, "Why is that?". By forming regular expressions or word graphs for the *Dialogue Manager* (DM) to parse, it is also possible to create a stimulus that matches a great number of user interactions. For example, the stimulus "*not*good*" matches both "Not so good" and "I'm not feeling that good".

Dialogue Management Tool

The *Dialogue Management Tool* (DMT) is a tool that aims to simplify the construction and maintenance of dialogues significantly. When constructing a dialogue, the tool makes crosschecks regarding types, names and quantity. It also maintains the consistency when updating the dialogues at a later state. Furthermore, it provides a time efficient way of creating dialogues, since the underlying structure does not have to be considered.

The DMT uses the new markup language *Dialogue Management Tool Language* (DMTL) in order to represent the dialogue and its states as a network (Gustavsson, Strindlund & Wiknertz, 2001).

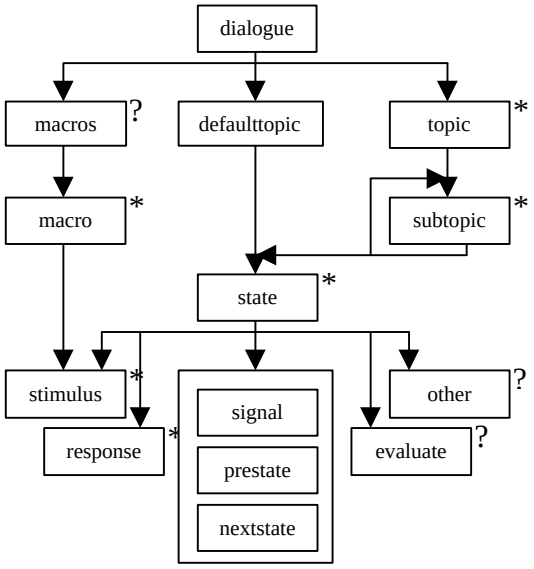


Figure 2. The structure of DMTL.

DMTL is an XML-based language and uses a *Document Type Definition* (DTD). A DTD is a set of rules that defines the grammar of an XML document. A document that fulfills the grammar rules in a specific DTD is called a *valid* document (Navarro, White & Burman, 2000). The output from the DMT is a valid DMTL document to be parsed by a DM. The structure of the DMTL DTD is shown in figure 2.

In order to give an overview of a dialogue, the previous conversation example between a TH and Anna will be expanded and step-by-step marked up according to the DMTL DTD.

The root element in DMTL is *dialogue*, which includes zero or one *macros*, zero or one *defaulttopic* and zero or more *topics*. A *macros* element includes zero or more *macro* elements that will be described later. The *defaulttopic* contains zero or more states, which cater for all the user inputs that do not match any other stimulus.

```
<dialogue>
    <defaulttopic> … </defaulttopic>
    <topic name="greeting"> … </topic>
</dialogue>
```

A *topic* includes zero or more *subtopics*.

```
<topic name="greeting">
    <subtopic name="casual"> … </subtopic>
    <subtopic name="polite"> … </subtopic>
</topic>
```

A *subtopic* in turn includes zero or more *subtopics* and zero or more *states*.

```
<subtopic name="casual">
    <subtopic name="swedish"> … </subtopic>
    <state name="initial" type="active"> …</state>
</subtopic>
```

A state includes stimuli, responses, prestates, nextstates, signals, evaluate and other.

The stimuli can be of several different types depending on the application; text, audio, visual and haptic, although text is the default value. For example, Anna might only look sad instead of saying "Not so good" in order to give a corresponding visual stimulus.

The responses could be plain text or marked up in any language. For example, the question/answer structure in a FAQ file could be maintained by using stimuli and responses. The response could also be marked up to direct or control the way in which the response is presented, for example, by using HTML anchors.

Prestate specifies the states from which the dialogue could have come and nextstate the states to which the dialogue can move.

The signal element enables the match to generate or emit a signal or notification to the DM, which it may choose to ignore or handle in some way. In the case example given, when Anna says: "I have to go. Goodbye!" the DM may simply close the connection.

The evaluate element can be used for defining a condition that has to be fulfilled before the dialogue is able to move into this particular state, hence this will increase the efficiency when searching the dialogue structure. For example, a variable can be set to imply that a state is visited and this can then be used as a condition for traversing another state. Other can be used for specifying any additional application specific information necessary or simply for adding comments. Though, the simple dialogue with Anna does not require evaluate or other.

The DMTL dialogue below describes the example given about the TH and Anna, thus it only constitutes a fragment of the whole dialogue.

```
<dialogue>
    <topic name="greeting">
        <subtopic name="casual">
            <state name="initial" type="active">
                <response>How are you?</response>
                <nextstate name="greeting.casual.bad"/>
                <nextstate:name=
                                "greeting.casual.good"/>
            </state>
            <state name="bad" type="linked">
                <stimulus>*not*good*</stimulus>
                <response>Why is that?</response>
                <nextstate
                    name=" greeting.casual.headache"/>
            </state>
            <state name="headache" type="linked">
                <stimulus>*headache*</stimulus>
                <response>Have you taken aspirin?
                </response>
            </state>
            <state name="bye" type="entry">
                <stimulus>*bye*</stimulus>
                <signal name="exit"/>
            </state>
            …
        </subtopic>
    </topic>
</dialogue>
```

In the current version of DMT there are four different state types; *linked*, *entry*, *visitswitch* and *active*. An *active* state is a state that invokes a question, without having to be triggered by a stimulus. An *entry* state is a state that can be invoked any time during the dialogue if the stimulus matches. A *linked* state is connected to other states by using *nextstate* or *prestate*. A *visitswitch* state points to several other states and works in a similar way as a case statement in C or Java. Which state the dialogue should move into depends on, for example, if the state has been visited before.

Dialogues tend to grow fast and become large and complex, with many topics, subtopics and states. This becomes an efficiency problem when a dialogue manager has to parse all the different paths in the dialogue when searching for a suitable stimulus. To avoid this an attribute for the *subtopic* element was introduced, *keyword*. This makes it possible to specify a number of keywords for each subtopic and only if any of these match the user input the subtopic is parsed to find a state with suitable stimulus.

Further, when creating stimuli all different ways of giving a specific stimulus must be considered. Since the natural language is complex, there are many different ways to express the same question. In order to facilitate for the user of the DMT, *macros* can be created to match the semantic of a certain stimulus. For example, the macro "WHATIS" can be used as "WHATIS VHML". This matches "What is VHML?", "What does VHML mean?" and so on.

Responses can be any text, but the current version of the DMT supports the *Virtual Human Markup Language* (VHML, 2001) within the text. Though, any markup language can be used in the dialogue. VHML is an XML-based language and is used for controlling the characters in a Virtual Human application, regarding sounds, emotions and movements of the body and in the face. Therefore, VHML can be useful when controlling the output of a TH application.

Since VHML, as well as DMTL, is an XML-based language, a problem exists in that the DMTL documents include VHML elements inside the *responses*. Because the VHML elements are not, and should not be, included in the DMTL DTD, the DMTL document will not be valid if the VHML elements remain inside the responses. The solution to this was to implement a transform function that transforms the VHML elements into plain text by using the standard entities for XML, i.e.:

Character	Entity
&	&
<	<
>	>
”	&quote;
’	'

For example:

```
<response>
    <vhml xml:lang=”en”>
        <p>
            <sad> Why is that? </sad>
        </p>
        </vhml>
</response>
```

is transformed into

```
<response>
    &lt;vhml xml:lang=&quote;en&quote;&gt;
        &lt;p&gt;
            &lt;sad&gt; Why is that? &lt;/sad&gt;
        &lt;/p&gt;
    &lt;/vhml&gt;
</response>
```

However, inside the *vhml* element these standard entities may already be used, which shows another problem. If, for example, a greater than sign is needed in the response, the user has to type in the standard entity > instead of >, as in any other XML document. The > is then transformed into plain text.

For example:

```
<response>
    <vhml>
        <p>
            5 &gt; 3
        </p>
    </vhml>
</response>
```

is transformed into

```
<response>
    &lt;vhml&gt;
        &lt;p&gt;
            5 &amp;gt; 3
        &lt;/p&gt;
    &lt;/vhml&gt;
</response>
```

To process an XML document, like the DMTL document, an API has to be used. There are two major types of XML APIs, tree-based APIs and event-based APIs. A tree-based API compiles an XML document into an internal tree structure and then allows an application to navigate that tree. The *Document Object Model* (DOM) is a standard tree-based API for XML and HTML documents, developed by the World Wide Web Consortium. An event-based API, on the other hand, reports parsing events, such as the start and end of elements, directly to the application through callbacks, and does not usually build an entire tree. The *Simple API for XML* (SAX), is an event-based API (SAX 2.0, 2001). SAX requires less memory than DOM and tends to run faster. However, with SAX, the application only sees the XML elements once and has to figure out what to do with the data right away, do it and then get ready to handle the next item. DOM, on the other hand, is more memory-intensive than SAX, since the entire document must be kept in memory at once. The advantage of DOM however, is that the application can go back and forth in the document and make changes to it (Navarro, White & Burman, 2000).

The input to the DMT is both saved as a DMTL document and stored as a DOM tree. The reason why DOM is used is that changes are made dynamically in a tree to update information at all times. The DMTL document keeps a static status of the DOM tree.

Future work

During the development of the DMT some issues have arisen that, if solved, will make the tool even more useful.

- The current version of DMT supplies VHML support by providing a list with VHML elements that can be inserted into the responses. To internationalize the DMT, this list should be written in the user’s language of choice.
- One useful feature would be to be able to import a file with another dialogue structure, not just DMTL, into the DMT. After updating, the file could be exported back to the original structure.
- In the DMT GUI, the states in a subtopic are presented in a list. When the user activates a state the information within this certain state is presented. It would be an advantage to be able to see the whole network or parts of the network graphically as well. This feature would provide the user with an even better overview of the dialogue.

Conclusions

The DMT makes construction of dialogues easier and keeps track of the state traversing in a conversation. Currently the DMT is based on responses marked up in VHML. An interactive detective story has been marked up in VHML using the DMT (Gustavsson, Strindlund & Wiknertz, 2001). This is only a small application, thus it constitutes a dialogue with approximately 500 states. Keeping track of these states is a complex task and shows the advantages of using a tool as DMT. Further, the current version of DMT has been found adequate with two other applications, the Mentor System developed by Marriott (to be published) and the FAQBot by Beard (1999). Other applications may require

alteration, but the current work shows a convenient means of constructing dialogues.

References

Beard, S. (1999), *FAQBot.* Honours thesis, Curtin University of Technology, Perth, Australia.

Gustavsson, C., Strindlund, L. & Wiknertz, E. (2001), *Verification, Validation and Evaluation of the Virtual Human Markup Language (VHML)*, Master thesis, Linköping University, Sweden.

Marriott, A. (to be published), 'A Java Based Mentor System'. In *Java in the Computer Science Curriculum*. Editor Greening, T. LNCS Springer.

Mauldin, M. L. (1994), 'Chatterbots, Tinymuds, And The Turing Test: Entering The Loebner Prize Competion'. In the proceedings of *AAAI-94*, AAAI Press, Seattle.

Navarro, A., White, C. & Burman, L. (2000), *Mastering XML*. SYBEX Inc., Alameda, CA.

SAX 2.0 (2001), *The Simple API for XML*. Available: http://www.megginson.com/SAX/index.html, [2001, August 10].

VHML (2001). *VHML*. Available: http://www.vhml.org, [2001, September 26]

Appendix C

VHML DTD

```
<!--
######################################################################
# Virtual Human Markup Language (VHML) DTD, version 0.4.             #
#                                                                    #
# Usage:                                                             #
# <!DOCTYPE vhml SYSTEM "http://www.vhml.org/vhml.dtd">              #
#                                                                    #
# Author: Camilla Gustavsson, c.gustavsson@home.se                   #
#         Linda Strindlund, linda.strindlund@home.se          `      #
#         Emma Wiknertz, wiknertz@home.se                            #
#                                                                    #
# Information about the VHML can be found at http://www.vhml.org     #
#                                                                    #
# Date: 15 November, 2001.                                           #
#                                                                    #
######################################################################
-->

<!--
########################################
# Some entities for an abstracter view #
########################################
-->

<!-- COMMENT:
New emotions are added here and specified below.
-->
<!ENTITY % EML
      "afraid |
       angry |
       confused |
       dazed |
       disgusted |
       happy |
       neutral |
       sad |
       surprised |
       default-emotion">

<!ENTITY % Emotion "( %EML; )">

<!-- COMMENT:
New gestures are added here and specified below.
-->
<!ENTITY % GML
      "agree |
       disagree |
       concentrate |
       emphasis |
       sigh |
       smile |
       shrug">

<!-- COMMENT:
New FAML elements are added here and specified below.
-->
<!ENTITY % FAML
      "look-left |
       look-right |
       look-up |
       look-down |
```

```
        eyes-left |
        eyes-right |
        eyes-up |
        eyes-down |
        head-left |
        head-right |
        head-up |
        head-down |
        head-roll-left |
        head-roll-right |
        eyebrow-up |
        eyebrow-down |
        eye-blink |
        wink |
        open-jaw |
        close-jaw">

<!-- COMMENT:
New SML elements are added here and specified below.
-->
<!-- COMMENT:
These elements are taken from SSML, Speech Synthesis Markup
Language.
Some more attributes to the elements are added.
http://www.w3.org/TR/speech-synthesis
-->
<!ENTITY % SML
       "break |
        emphasize-syllable |
        emphasise-syllable |
        phoneme |
        prosody |
        say-as |
        voice">

<!-- COMMENT:
New XHTML elements are added here and specified below.
-->
<!ENTITY % XHTML
       "a |
        anchor">

<!ENTITY % allowed-on-lower-level
      "(#PCDATA | mark | embed | %GML; | %FAML; | %SML; |
      %XHTML;)*">

<!-- COMMENT:
Can be a relative value or one of
low, medium or high.
-->
<!ENTITY % intensityvalue "CDATA">

<!ENTITY % targetname "CDATA">

<!ENTITY % sourcepath "CDATA">

<!ENTITY % integer "CDATA">

<!ENTITY % secs-or-msecs "CDATA">

<!ENTITY % id "CDATA">
```

```
<!ENTITY % substitute-string "CDATA">

<!ENTITY % phoneme-string "CDATA">

<!ENTITY % contour-format "CDATA"> <!-- from SSML -->

<!-- COMMENT:
Can be a relative change or one of
low, medium, high or default.
-->
<!ENTITY % pitchvalues "CDATA">

<!-- COMMENT:
Can be a relative change or one of
low, medium, high or default.
-->
<!ENTITY % rangevalues "CDATA">

<!-- COMMENT:
Can be a relative change or one of
slow, medium, fast or default.
-->
<!ENTITY % ratevalues "CDATA">

<!-- COMMENT:
Can be a relative change or one of
silent, soft, medium, loud or default.
-->
<!ENTITY % volumevalues "CDATA">

<!ENTITY % voice-name-list "CDATA"> <!-- from SSML -->

<!ENTITY % link-type-list "CDATA">

<!ENTITY % character-list "CDATA">

<!ENTITY % uri "CDATA">

<!ENTITY % coordinate-list "CDATA">

<!ENTITY % script "CDATA">

<!ENTITY % say-as-types
      "(acronym | number | number:ordinal | number:digits |
        date | date:dmy | date:mdy | date:ymd | date:ym |
        date:my | date:md | date:y | date:m | date:d | time |
        time:hms | time:hm | time:h | duration | duration:hms |
        duration:hm | duration:ms | duration:h | duration:m |
        duration:s | currency | measure | telephone | name |
        net | net:email | net:uri | address )">
        <!-- from SSML -->

<!ENTITY % default-EML-attributes
      "duration %secs-or-msecs; #IMPLIED
       intensity %intensityvalue; 'medium'
       mark %id; #IMPLIED
       wait %secs-or-msecs; #IMPLIED">

<!ENTITY % default-GML-attributes
      "%default-EML-attributes;">
```

```
<!ENTITY % default-FAML-attributes
      "%default-EML-attributes;">

<!ENTITY % default-XHTML-attributes
      "accesskey %id; #IMPLIED
       coords %coordinate-list; #IMPLIED
       onblur %script; #IMPLIED
       onfocus %script; #IMPLIED
       shape (default | rect | circle | poly) #IMPLIED
       tabindex %integer; #IMPLIED">
       <!-- The tabindex must be between 0 and 32,767 -->


<!--
####################
# Elements in VHML #
####################
-->

<!ELEMENT vhml (paragraph | p | person | mark)+>
<!ATTLIST vhml
      xml:lang NMTOKEN #IMPLIED>

<!ELEMENT person (paragraph | p | mark)*>
<!ATTLIST person
      age %integer; #IMPLIED
      category (child | teenager | adult | elder) #IMPLIED
      gender (female | male | neutral) #IMPLIED
      name %voice-name-list; #IMPLIED
      variant %integer; #IMPLIED
      disposition %Emotion; #IMPLIED>

<!ELEMENT paragraph (#PCDATA | mark | embed | %EML; | %GML; |
                     %FAML; | %SML; | %XHTML;)*>

<!ATTLIST paragraph
      xml:lang NMTOKEN #IMPLIED
      target %targetname; #IMPLIED>

<!ELEMENT p (#PCDATA | mark | embed | %EML; | %GML; | %FAML;
             | %SML; | %XHTML;)*>
<!ATTLIST p
      xml:lang NMTOKEN #IMPLIED
      target %targetname; #IMPLIED>

<!ELEMENT mark EMPTY>
<!ATTLIST mark
      name CDATA #REQUIRED>

<!ELEMENT embed EMPTY>
<!ATTLIST embed
      type (audio | mml) #REQUIRED
      src %sourcepath; #REQUIRED>
```

```
<!--
###################
# Elements in EML #
###################
-->

<!ELEMENT afraid %allowed-on-lower-level;>
<!ATTLIST afraid
      %default-EML-attributes;>

<!ELEMENT angry %allowed-on-lower-level;>
<!ATTLIST angry
      %default-EML-attributes;>

<!ELEMENT confused %allowed-on-lower-level;>
<!ATTLIST confused
      %default-EML-attributes;>

<!ELEMENT dazed %allowed-on-lower-level;>
<!ATTLIST dazed
      %default-EML-attributes;>
<!ELEMENT disgusted %allowed-on-lower-level;>
<!ATTLIST disgusted
      %default-EML-attributes;>

<!ELEMENT happy %allowed-on-lower-level;>
<!ATTLIST happy
      %default-EML-attributes;>

<!ELEMENT neutral %allowed-on-lower-level;>
<!ATTLIST neutral
      %default-EML-attributes;>

<!ELEMENT sad %allowed-on-lower-level;>
<!ATTLIST sad
      %default-EML-attributes;>

<!ELEMENT surprised %allowed-on-lower-level;>
<!ATTLIST surprised
      %default-EML-attributes;>

<!--COMMENT:
This is for the default emotion in the person element if there is
one. Otherwise the system default emotion will be used
-->
<!ELEMENT default-emotion %allowed-on-lower-level;>
<!ATTLIST default-emotion
      %default-EML-attributes;>


<!--
###################
# Elements in GML #
###################
-->

<!ELEMENT agree %allowed-on-lower-level;>
<!ATTLIST agree
      %default-GML-attributes;
      repeat %integer; '1'>
```

```
<!ELEMENT disagree %allowed-on-lower-level;>
<!ATTLIST disagree
      %default-GML-attributes;
      repeat %integer; '1'>

<!ELEMENT concentrate %allowed-on-lower-level;>
<!ATTLIST concentrate
      %default-GML-attributes;>

<!ELEMENT emphasis %allowed-on-lower-level;>
<!ATTLIST emphasis
      %default-GML-attributes;
      level (reduced | none | moderate | strong) 'moderate'>
<!ELEMENT sigh %allowed-on-lower-level;>
<!ATTLIST sigh
      %default-GML-attributes;
      repeat %integer; '1'>

<!ELEMENT smile %allowed-on-lower-level;>
<!ATTLIST smile
      %default-GML-attributes;>

<!ELEMENT shrug %allowed-on-lower-level;>
<!ATTLIST shrug
      %default-GML-attributes;
      repeat %integer; '1'>


<!--
##################
# Element in SML #
##################
-->

<!ELEMENT break EMPTY>
<!ATTLIST break
      mark %id; #IMPLIED
      size (none | small | medium | large) 'medium'
      time %secs-or-msecs; #IMPLIED
      smooth (yes | no) 'yes'>

<!ELEMENT emphasize-syllable (#PCDATA)>
<!ATTLIST emphasize-syllable
      mark %id; #IMPLIED
      target %phoneme-string; #IMPLIED
      level (reduced | none | moderate | strong) 'moderate'
      affect (pitch | duration | both) 'pitch'>

<!ELEMENT emphasise-syllable (#PCDATA)>
<!ATTLIST emphasise-syllable
      mark %id; #IMPLIED
      target %phoneme-string; #IMPLIED
      level (reduced | none | moderate | strong) 'moderate'
      affect (pitch | duration | both) 'pitch'>

<!ELEMENT phoneme (#PCDATA)>
<!ATTLIST phoneme
      mark %id; #IMPLIED
      alphabet (ipa | worldbet | xsampa) #IMPLIED
      ph %phoneme-string; #REQUIRED>
```

```
<!ELEMENT prosody %allowed-on-lower-level;>
<!ATTLIST prosody
      mark %id; #IMPLIED
      contour %contour-format; #IMPLIED
      duration %secs-or-msecs; #IMPLIED
      pitch %pitchvalues; 'default'
      range %rangevalues; 'default'
      rate %ratevalues; 'default'
      volume %volumevalues; 'default'>

<!ELEMENT say-as (#PCDATA)>
<!ATTLIST say-as
      mark %id; #IMPLIED
      type %say-as-types; #REQUIRED
      sub %substitute-string; #IMPLIED>

<!ELEMENT voice %allowed-on-lower-level;>
<!ATTLIST voice
      mark %id; #IMPLIED
      age %integer; #IMPLIED
      category (child | teenager | adult | elder) #IMPLIED
      gender (female | male | neutral) #IMPLIED
      name %voice-name-list; #IMPLIED
      variant %integer; #IMPLIED>


<!--
####################
# Elements in FAML #
####################
-->

<!ELEMENT look-left %allowed-on-lower-level;>
<!ATTLIST look-left
      %default-FAML-attributes;>

<!ELEMENT look-right %allowed-on-lower-level;>
<!ATTLIST look-right
      %default-FAML-attributes;>

<!ELEMENT look-up %allowed-on-lower-level;>
<!ATTLIST look-up
      %default-FAML-attributes;>

<!ELEMENT look-down %allowed-on-lower-level;>
<!ATTLIST look-down
      %default-FAML-attributes;>

<!ELEMENT eyes-left %allowed-on-lower-level;>
<!ATTLIST eyes-left
      %default-FAML-attributes;>

<!ELEMENT eyes-right %allowed-on-lower-level;>
<!ATTLIST eyes-right
      %default-FAML-attributes;>

<!ELEMENT eyes-up %allowed-on-lower-level;>
<!ATTLIST eyes-up
      %default-FAML-attributes;>
```

```
<!ELEMENT eyes-down %allowed-on-lower-level;>
<!ATTLIST eyes-down
      %default-FAML-attributes;>

<!ELEMENT head-left %allowed-on-lower-level;>
<!ATTLIST head-left
      %default-FAML-attributes;>

<!ELEMENT head-right %allowed-on-lower-level;>
<!ATTLIST head-right
      %default-FAML-attributes;>

<!ELEMENT head-up %allowed-on-lower-level;>
<!ATTLIST head-up
      %default-FAML-attributes;>

<!ELEMENT head-down %allowed-on-lower-level;>
<!ATTLIST head-down
      %default-FAML-attributes;>

<!ELEMENT head-roll-left %allowed-on-lower-level;>
<!ATTLIST head-roll-left
      %default-FAML-attributes;>

<!ELEMENT head-roll-right %allowed-on-lower-level;>
<!ATTLIST head-roll-right
      %default-FAML-attributes;>

<!ELEMENT eyebrow-up %allowed-on-lower-level;>
<!ATTLIST eyebrow-up
      %default-FAML-attributes;
      which (both | left | right) 'both'>

<!ELEMENT eyebrow-down %allowed-on-lower-level;>
<!ATTLIST eyebrow-down
      %default-FAML-attributes;
      which (both | left | right) 'both'>

<!ELEMENT eye-blink EMPTY>
<!ATTLIST eye-blink
      %default-FAML-attributes;
      repeat %integer; '1'>

<!ELEMENT wink EMPTY>
<!ATTLIST wink
      %default-FAML-attributes;
      which (left | right) 'left'
      repeat %integer; '1'>

<!ELEMENT open-jaw %allowed-on-lower-level;>
<!ATTLIST open-jaw
      %default-FAML-attributes;>

<!ELEMENT close-jaw %allowed-on-lower-level;>
<!ATTLIST close-jaw
      %default-FAML-attributes;>
```

```
<!--
#####################
# Elements in XHTML #
#####################
-->

<!ELEMENT a (#PCDATA)>
<!ATTLIST a
      %default-XHTML-attributes;
      charset %character-list; #IMPLIED
      href %uri; #IMPLIED
      hreflang NMTOKEN #IMPLIED
      name %id; #IMPLIED
      rel %link-type-list; #IMPLIED
      rev %link-type-list; #IMPLIED
      type NMTOKEN #IMPLIED>

<!ELEMENT anchor (#PCDATA)>
<!ATTLIST anchor
      %default-XHTML-attributes;
      charset %character-list; #IMPLIED
      href %uri; #IMPLIED
      hreflang NMTOKEN #IMPLIED
      name %id; #IMPLIED
      rel %link-type-list; #IMPLIED
      rev %link-type-list; #IMPLIED
      type NMTOKEN #IMPLIED>
```

Appendix D

DMTL DTD

```
<!--
#########################################################################
# Dialogue Manager Tool Markup Language (DMTL) DTD, version 4.0.  #
#                                                                       #
# Usage:                                                                #
# <!DOCTYPE dialogue SYSTEM "http://www.vhml.org/DTD/dmtl.dtd">   #
#                                                                       #
# Author: Camilla Gustavsson, c.gustavsson@home.se                      #
#         Linda Strindlund, linda.strindlund@home.se                    #
#         Emma Wiknertz, wiknertz@home.se                               #
#                                                                       #
# Date: 17 October 2001                                                 #
#                                                                       #
#########################################################################
-->

<!ELEMENT dialogue (macros?, defaulttopic?, topic*)>

<!ELEMENT macros (macro)*>

<!ELEMENT macro (stimulus)*>
<!ATTLIST macro
      name CDATA #REQUIRED>


<!-- COMMENT:
# This is used to specify a default answer that triggers if
# there is no other answers matching the stimulus
-->
<!ELEMENT defaulttopic (state)+>


<!ELEMENT topic (subtopic)*>
<!ATTLIST topic
      name CDATA #REQUIRED>


<!ELEMENT subtopic (state | subtopic)*>
<!ATTLIST subtopic
      name CDATA #REQUIRED
      keywords CDATA #IMPLIED
      evaluate CDATA #IMPLIED>


<!-- COMMENT:
# It has a type to cater for the different types of nodes that may
# need to be specified (for example, some nodes may be "active" –
# that is, the Dialogue Manager which uses this file may use an
# "active" node to ask the user questions or make observations, not
# just respond to stimulus.
# linked - the stimulus is matched only from "nextstates"
# active - pro-active interaction with the user.
# entry  - these stimuli are used for initial input from user
# switch - the start of a chained stimulus-response set of states
#          to cater for learned behaviour in the user.
-->
<!ELEMENT state ( stimulus*,
                  response*,
                  (prestate*,nextstate*,signal*)*,
                  evaluate?,
                  other?) >
```

```
<!ATTLIST state
      name CDATA #REQUIRED
      type (linked | active | entry | visitswitch ) "entry">


<!-- COMMENT:
# The stimulus is typically a question or a response to a question
# or could be input from a facial recognition system - a shrug, a
# nod, etc. Case is important in the input.
-->
<!ELEMENT stimulus (#PCDATA)>
<!ATTLIST stimulus
      type (text | visual | audio | haptic) "text">


<!-- COMMENT:
# The response is typically a response but marked up in vhml. The
# response could be text, XHTML text, text plus EML, etc. The
# response could also be a question for pro-active dialogues. The
# vhml does not have the vhml root tag.
#
# The response weight is a floating point number between 0.0 and
# 1.0 with 0.0 meaning no confidence in this response and 1.0
# meaning total confidence in the response. A value of 0.7 could be
# the typical value for most responses which match. This gives
# the ability to have other responses match but at a higher
# priority because the response is seen as being more important in
# this situation. The default value for a response weight is 0.7.
# The Dialogue Manager may ignore this value.
-->
<!ELEMENT response (#PCDATA)>
<!ATTLIST response
      weight CDATA "0.7"
      statereference CDATA #IMPLIED>

<!-- COMMENT:
# A signal tag enables the match to generate a signal or a
# notification to the Dialogue Manager which it may choose to
# ignore. An example of the use of this is if the match has
# determined that the user wants to finish the dialogue and hence
# the DM should know to finish. The value of the signals should be
# one of a set of descripted values instead of just CDATA, but
# these values are not yet decided which to be.
-->
<!ELEMENT signal EMPTY>
<!ATTLIST signal
      name CDATA #REQUIRED>


<!-- COMMENT:
# The prestate tag specifies a set of states which must match for
# this state to match the stimulus. This allows for catering for a
# specific "yes" answer but only to the prestate question.
-->

<!ELEMENT prestate EMPTY>
<!ATTLIST prestate
      name CDATA #REQUIRED>
```

```
<!-- COMMENT:
# The nextstate tag specifies a set of states to test for follow-up
# stimulus input. These states would be checked first (perhaps with
# some increase in the response weighting?) before all other
# states. This allows for catering for a specific "yes" answer to
# this response.
-->
<!ELEMENT nextstate EMPTY>
<!ATTLIST nextstate
      name CDATA #REQUIRED>


<!-- COMMENT:
# The evaluate tag specifies different application specific test
# that has to be made. If the contained data begins with // it
# means that it is a comment.
-->
<!ELEMENT evaluate (#PCDATA)>


<!-- COMMENT:
# The other tag gives the opportunity to specify other application
# specific information.
-->
<!ELEMENT other (#PCDATA)>
```

Appendix E

User manual

User manual

The main objective of the DMT is that it should be a useful tool when creating and maintaining dialogues. These dialogues can be included when developing, for example, an interactive Talking Head application or an ordinary Question/Answer file.

The dialogue structure

In order to structure a dialogue, a network is used. The overall structure of a dialogue is shown in figure 1.

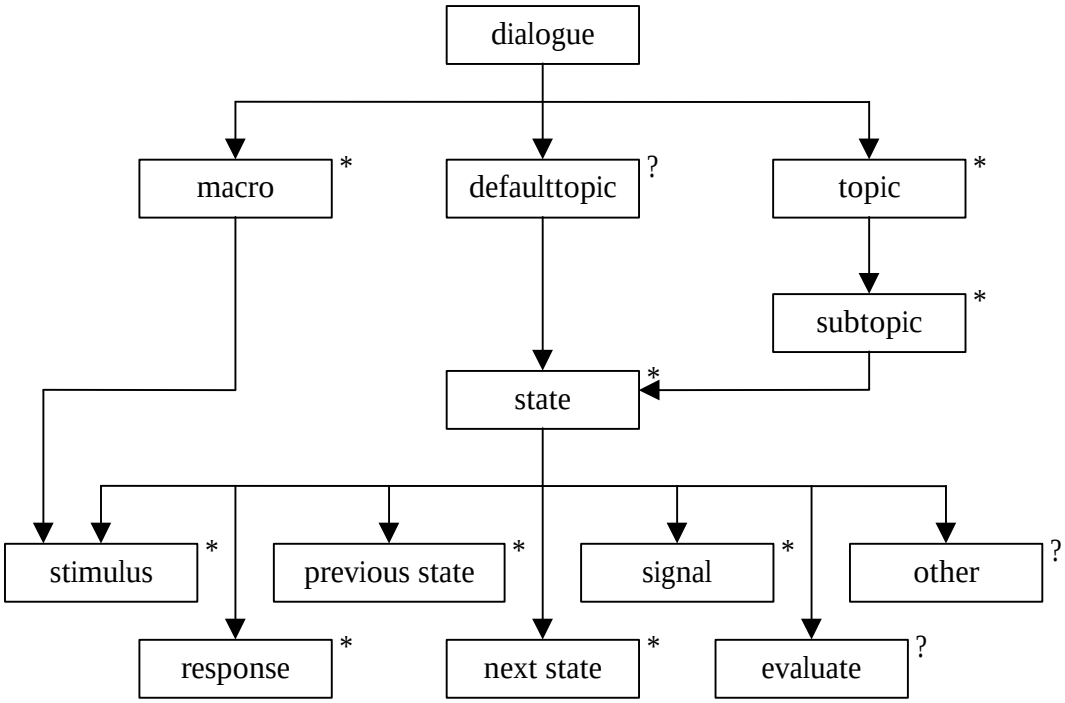


Figure 1. The structure of a dialogue.

An arrow from A to B means that A can consist of B. The number of B's is specified using stars and question marks. A star (*) after the box means that it can occur zero or more times. A question mark (?) indicates that it can occur zero or one time.

The Graphical User Interface

The GUI is divided into six different parts. These are the **Menubar**, the **Toolbar**, the **Subtopic path**, the **State list**, the **State information** and the **Error status.**

A screen shot of the Graphical User Interface (GUI) is shown in figure 2.

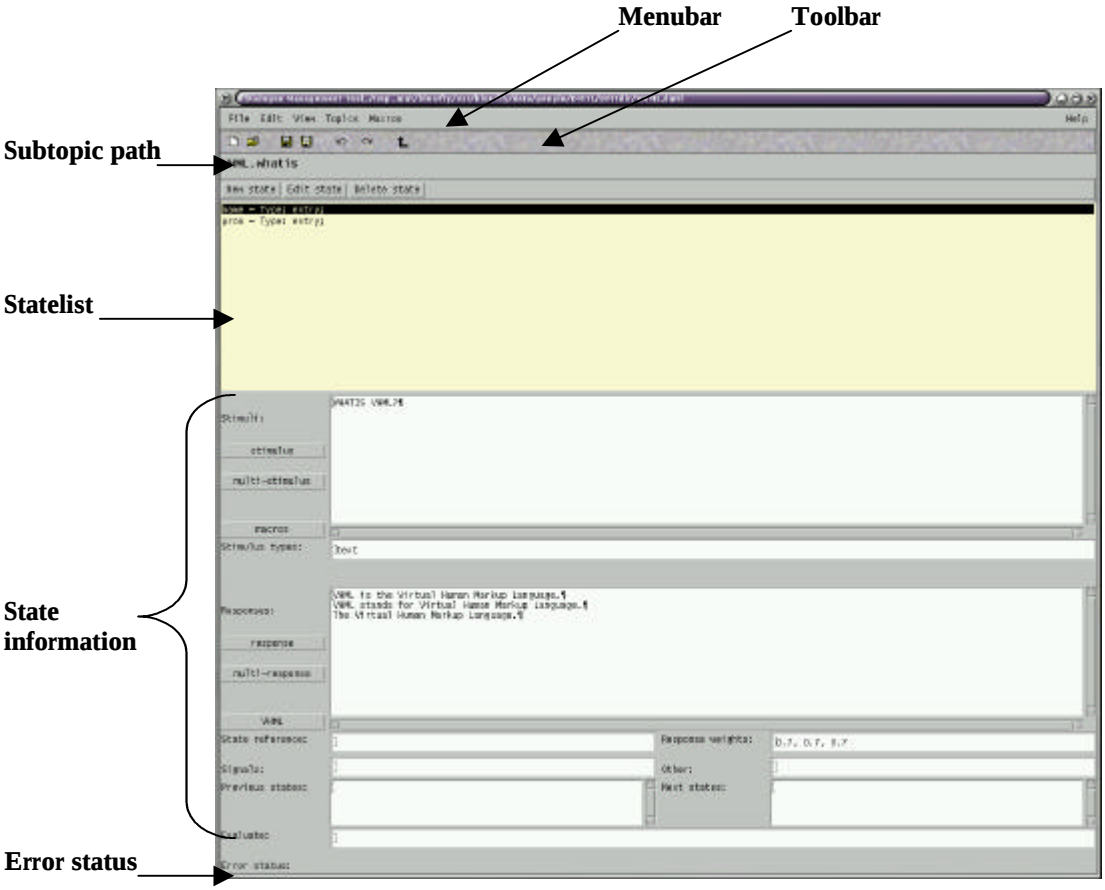
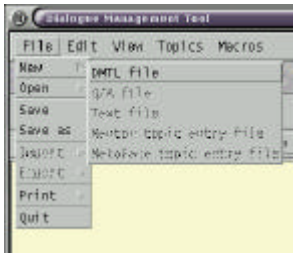


Figure 2. The GUI of the DMT.

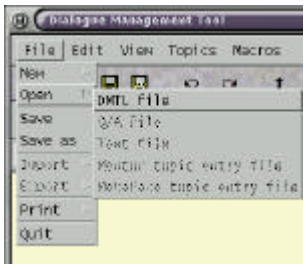
New file



There are two options when opening a new file. The first one is to use the **File** menu in the **Menubar** and select **New** and then **DMTL file**. The second way is to click the **New** image in the **Toolbar**. If the current DMTL file is not saved, you will be asked whether to save it or not before opening a new file, since opening a new file will lead to that the current file will be closed.

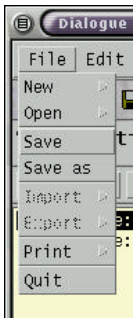
When starting the DMT a new file will automatically be opened.

Open file



There are two options when opening an existing file. The first one is to use the **File** menu in the **Menubar** and select **Open** and then **DMTL file**. The second way is to click the **Open** image in the **Toolbar**. If the current DMTL file is not saved, you will be asked whether to save it or not before opening an existing file, since opening another file will lead to that the current file will be closed.

Save file



There are two options when saving a file. The first one is to use the **File** menu in the **Menubar** and select **Save**. The second way is to click the **Save** image in the **Toolbar**.

If the file has not been saved before and hence has not got a name yet you will be asked to type in a file name. Another way is to select **Save as** under the **File** menu in the **Menubar** or to click the **Save as** image in the **Toolbar**. This can be done with either an unnamed file or to provide a named file with a new name, hence make a copy of the original file.

Quit DMT

To quit the DMT select **Quit** under the **File** menu in the **Menubar**. If the current DMTL file is not saved, you will be asked whether to save it or not before quitting DMT.

Undo

It is only possible to undo changes in the fields in the **State information** area. However, only the ten most recent changes can be undone. To undo the most recent changes in the current DMTL file, select **Undo** from the **Edit** menu in the **Menubar** or click the **Undo** image in the **Toolbar**.

Redo

It is only possible to redo changes in the fields in the **State information** area. However, only the ten most recent changes can be redone. To redo the most recent changes that have been undone in the current DMTL file, select **Redo** from the **Edit** menu in the **Menubar** or click the **Redo** image in the **Toolbar**.

Macros

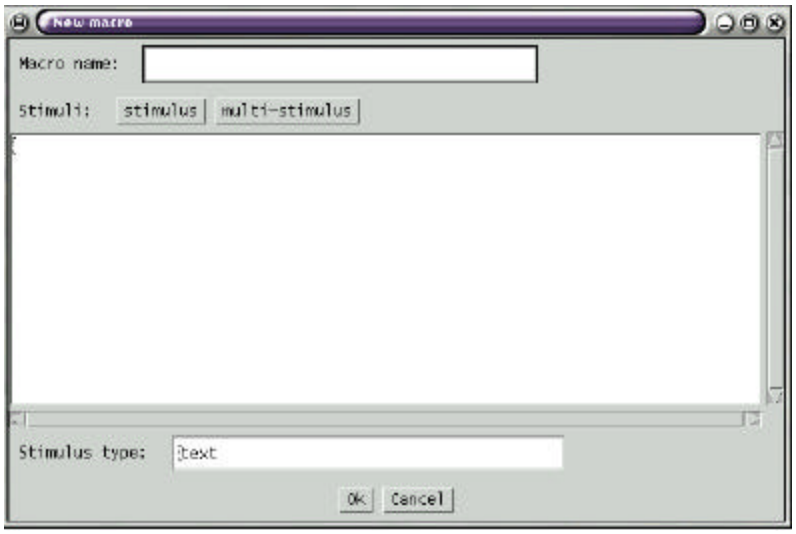
When creating stimuli, all different ways of specifying a particular stimulus must be considered. Since the natural language is complex, there are many different ways to express the same question. Macros can be created to match the semantic of a certain stimulus. For example, the macro “WHATIS” can be used in the sentence “WHATIS VHML?”. This would match “What is VHML?”, “Can you please tell me about VHML? and so on. In order to differ from ordinary text in the stimulus, the macro names are in uppercase.

It could also be useful to have parameters for the macros. One way of doing that is to use parenthesizes and brackets. An example of this is “WHATIS(VHML)” or “WHATIS(a DTD)”. The parameter is “VHML” respective “a DTD”. The parameter can be any text string or even a macro itself. Then inside the stimuli in the macro mark the place in the sentence where the parameter should be placed with brackets.

The macros are presented in the **Macros** menu in the **Menubar** when the DMTL file is opened. The menu can be torn off, by clicking the dotted line at the top of the menu, and placed wherever on the screen you find suitable. This gives a better overview of the macros included in the DMTL file.

New macro

To create a new macro, select **New** from the **Macros** menu in the **Menubar** and a dialogue box will appear on the screen.



Type in a name in the **Name** field. The name must be in uppercase to differ from plain text. Further, it has to be unique, i.e. there can not exist two macros with the same name in a DMTL file.

In the **Stimuli** field, type in the different stimuli that the macro should expand to. Use the **stimulus** and **multi-stimulus** buttons above the **Stimuli** field to mark it as zero or more stimuli. The **stimulus** button sets a *stimulus mark* (¶) in the position of the mark, so make certain that the mark is placed after the stimulus. A way to create more than one stimulus is to type in a number of stimulus in the field, one on each row. Then highlight all the stimuli and click the **multi-stimulus** button. In this way, a stimulus mark will be inserted at the end of each row, making each row a separate stimulus.

When all stimuli have been created, the types of the different stimuli have to be decided. A stimulus can be of several different types depending on the application; *text*, *audio*, *visual* and *haptic*, although *text* is the default value. For example, instead of having "Yes" as a text stimulus, there can be a visual stimulus when the user nods. Since *text* is the default type, it is already specified in the **Stimulus types** field. If the same type is wanted for all stimuli, one type is enough in the field, though every stimulus will get the specified type. If different types are demanded, one type for each stimulus has to be typed in, in the same order as the stimuli.

When the name, stimuli and stimulus types are typed in, click the **Ok** button to create the new macro or the **Cancel** button to return to the DMT without creating a macro.

Edit macro

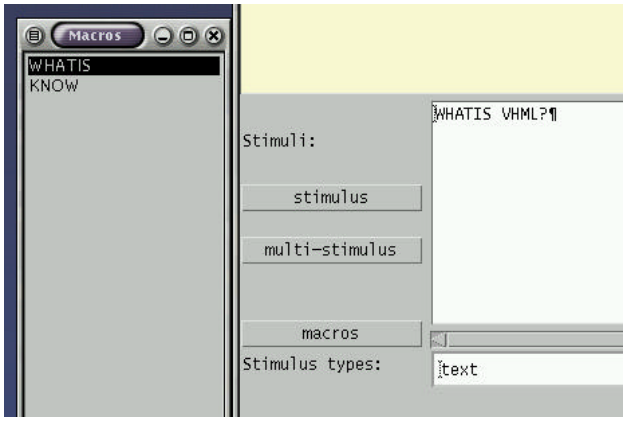
To edit a macro, go to the **Macros** menu in the **Menubar** and select the macro to edit, then select **Edit**. A similar dialogue box as for creating a new macro will appear on the screen, but with the current information about the macro inserted to the fields. To edit the macro, change the information in the fields in the same way as described in section New macro.

Then click the **Ok** button to keep the changes or the **Cancel** button to return to the DMT without changes.

Delete macro

To delete a macro, go to the **Macros** menu in the **Menubar** and select the macro to delete, then select **Delete**. A confirming dialogue box will appear on the screen. If you want to proceed, click the **Ok** button, if not, click the **Cancel** button.

Use macro



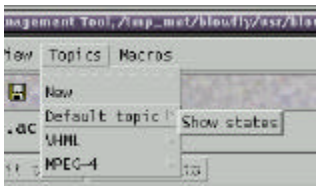
Inserting a macro to a certain stimulus can be made in two different ways. Firstly, you can type in the macro by hand in the **Stimuli** field in the **State information** area. By using this method, you have to make sure that the macro is in uppercase and spelled correctly. If not, the macro will be treated as plain text and hence not give you the demanded functionality. Another, more secure, way to insert the macros is by using the **Macros list**. The **Macros list** can be opened by clicking the **macros** button on the left hand side of the **Stimuli** field. To insert a certain macro, select the macro in the **Macros list** by using the mouse or the arrow keys on the keyboard. When the wanted macro is selected, insert it to the **Stimuli** field by double clicking it or using the enter key on the keyboard.

Default topic

When opening a new file the file is completely empty except from a defaulttopic with one state. The state has a stimulus that matches everything and the response “Sorry, but I can’t help you with that”. When opening an existing file that does not include any defaulttopic, the same defaulttopic is inserted automatically.

The defaulttopic caters for all the user input that does not match any other stimulus. The defaulttopic can contain zero or more states and hence gives the user a possibility to have many different default responses. This can be useful when having responses such as “ Sorry, but I can’t understand that.” or “Sorry, I don’t know that person.”. The first response can be used as a default response to everything. The second response is used when the dialogue manager know that the stimulus is about a person, but it has no information about that person The idea with defaulttopic gives the user a possibility to design these default responses in a certain way, best suitable for their specific application.

Show default states



To show the current states in the defaulttopic included in the DMTL file, select **Show states** under **Default topic** in the **Topics** menu in the **Menubar**. The included default states will be presented in the **State list**, section Show states.

New default state

A new default state is created in the same way as an ordinary state, section New state.

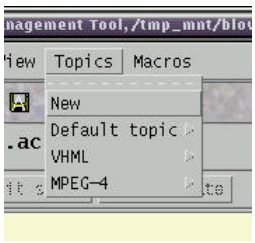
Edit default state

A default state is edited in the same way as an ordinary state, section Edit state.

Delete default state

A default state is deleted in the same way as an ordinary state, section Delete state.

Topic

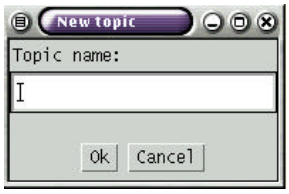


A topic includes zero or more subtopics. The topic has a name that is an identifier to that specific topic. By using topics, the structure of the dialogue becomes organized and well presented.

The topics are presented in the **Topics** menu in the **Menubar** when a DMTL file is opened. The menu can be torn off, by clicking on the dotted line at the top of the menu, and placed wherever on the screen you find suitable. This gives a better overview of the topics included in the DMTL file.

New topic

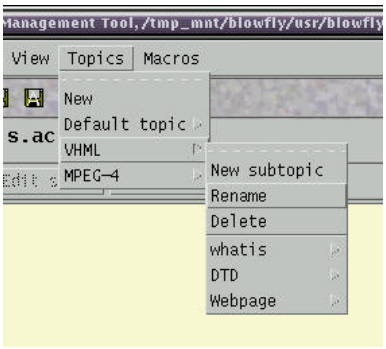
To create a new topic, select **New** in the **Topics** menu in the **Menubar**. When this action is performed a dialogue box will appear on the screen.



Type in a name in the **Topic name** field. The name is prohibited to contain any dots (.) or commas (,). Further, the name has to be unique within the set of topics.

When a correct name is typed in, click the **Ok** button to create the new topic or on the **Cancel** button to return to the DMT without creating a topic.

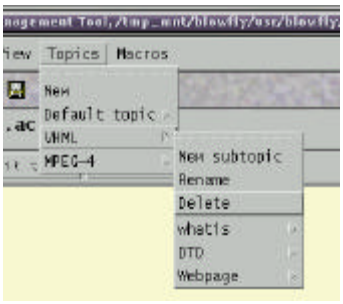
Rename topic



To rename a topic, go to the **Topics** menu in the **Menubar** and select the topic to rename, then select **Rename**. A similar dialogue box as for creating a new topic will appear on the screen, but with the current name of the topic inserted into the **Topic name** field. To rename, the topic change the information in the **Topic name** field in the same way as described in section New topic.

Then click the **Ok** button to keep the changes or the **Cancel** button to return to the DMT without changes.

Delete topic



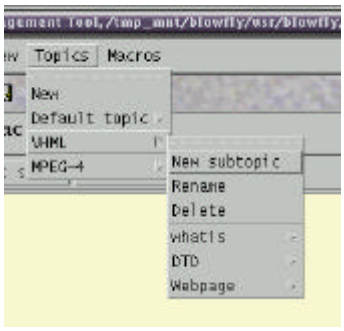
To delete a topic, go to the **Topics** menu in the **Menubar** and select the topic to delete, then select **Delete**. A confirming dialogue box will appear on the screen. If you want to proceed, click the **Ok** button, if not, click the **Cancel** button.

By deleting a topic you should be aware of that you also delete all references pointing to states in that topic. Read more about different references in sections Responses, Previous states and Next states.

Subtopic

A subtopic includes zero or more subtopics and zero or more states. The subtopic has a name that is an identifier to that specific subtopic.

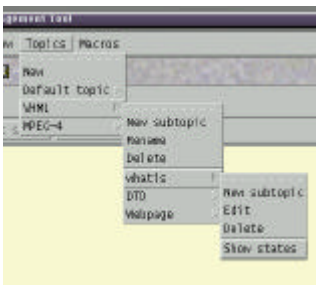
Dialogues tend to grow fast and become large and complex, with many topics, subtopics and states. This becomes an efficiency problem when a dialogue manager has to parse all the different paths in the dialogue while searching for a suitable stimulus. To avoid this, keywords are used. This makes it possible to specify a number of keywords for each subtopic and only if any of these match the user input the subtopic is parsed to find a suitable stimulus.



Yet another way to decrease the numbers of paths to parse is to use an evaluate statement for the subtopics. With evaluate some conditions can be set and these has to be fulfilled in order to parse that specific subtopic.

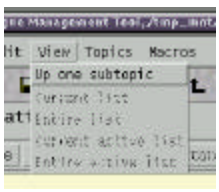
The subtopics are presented in the **Topics** menu in the **Menubar** under their respective topic. The menu can be torn off, by clicking on the dotted line at the top of the menu, and placed wherever on the screen you find suitable. This gives a better overview of the subtopics included in a specific topic in the DMTL file.

Show states



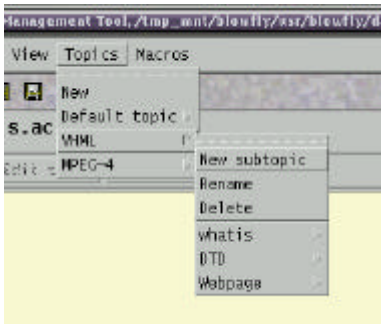
In order to view the states in a specific subtopic, first select the subtopic to be viewed. To do this, select the specific topic or subtopic in the **Topics** menu in the **Menubar**. When the subtopic is selected, select **Show states** under that subtopic. An easier way to show the states is to use the tear off menus, section Hints for the user. The states in the subtopic are presented in the **State list**. Each state is presented with information such as name, type, previous states, next states, signals, evaluate and other. Read more about states in section State. The path to the shown subtopic is presented in the **Subtopic path** above the **State list**. The path is a fully qualified name for the shown subtopic. A fully qualified name is a name that gives the whole search path to a subtopic. For example, a subtopic called *whatis* in a topic *VHML*, has the fully qualified name *VHML.whatis.*

Up a subtopic



Since a subtopic can contain other subtopics, it is possible to move up one level in the dialogue and show the states on the level above. This is done either by selecting **Up a subtopic** from the **View** menu in the **Menubar** or by clicking the **Up** image in the **Toolbar**.

New subtopic



A new subtopic is created by first selecting the topic or subtopic, in which to create the subtopic, from the **Topics** menu in the **Menubar**. To create a new subtopic, select **New subtopic** in that topic or subtopic. When this action is performed a dialogue box will appear on the screen.

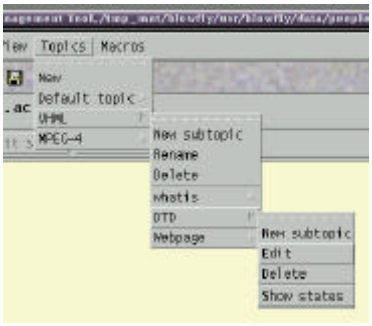
Type in a name in the **Subtopic name** field. The name can not contain any dots (.) or commas (,). Further, the name has to be unique within the set of subtopics in the particular topic or subtopic. A name is required, hence you can not leave this field blank.

Any keywords associated with this subtopic can be typed into the **Keywords** field. The keywords should be separated by commas (,).

A condition may be typed into the **Evaluate** field. You can read more about the format of the condition in section Evaluate.

When a correct name is typed in and the keywords and the evaluate condition are set, click the **Ok** button to create the new subtopic or the **Cancel** button to return to the DMT without creating a subtopic.

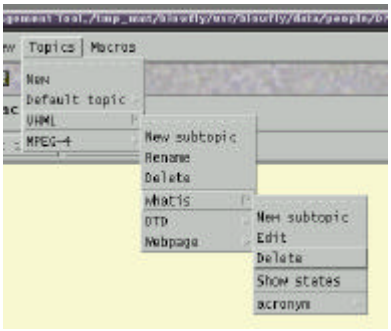
Edit subtopic



To edit a subtopic, go to the **Topics** menu in the **Menubar** and select the subtopic to edit, then select **Edit**. A similar dialogue box as for creating a subtopic will appear on the screen, but with the current information of the subtopic inserted to the fields. To edit the subtopic, change the information in the fields in the same way as described in section New subtopic.

Then click the **Ok** button to keep the changes or the **Cancel** button to return to the DMT without changes.

Delete subtopic



To delete a subtopic, go to the **Topics** menu and select the subtopic to delete, then select **Delete**. A confirming dialogue box will appear on the screen. If you want to proceed, click the **Ok** button, if not, click the **Cancel** button.

By deleting a subtopic you should be aware of that you also delete all references pointing to states in that subtopic. Read more about references in the sections Responses, Previous states and Next states.

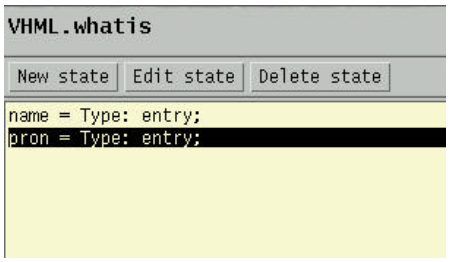
State

A state includes stimuli, responses, previous states, next states, signals, evaluate and other. A state also has a name that works as an identifier for the specific state and a type that determines the functionality of the state.

In the current version of DMTL there are four different state types:

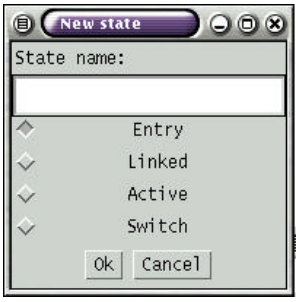
- *active*. A state that invokes a question, without having to be triggered by a stimulus. For example, the question “Do you want to know more about VHML?”.
- *entry*. A state that can be invoked at any time during the dialogue, if the stimulus matches. This is also the default state type. An example of this is “What is VHML?”.
- *linked*. A state that is connected to other states by using next states or previous states. The state is linked because the stimulus depends on having some kind of context. An example is the question “What is that?”, where “that” corresponds to something introduced earlier in the conversation and the dialogue manager should know what it is. A linked state can never directly match the initial user input, it has to be linked from another state.
- *visitswitch*. A state that points to several other states and works in a similar way as a case statement in C or Java. Which state the dialogue should move into can, for example, depend on if the state has been visited before. If a state is being visited that state is marked as visited. The *visitswitch* specifies the priority order in which the states should be moved into, but makes certain that no state is visited more than once.

New state



In order to create a new state you must select a subtopic and then select **Show states** for that specific subtopic, see section Show states. Then click the **New state** button above the **State list**. When this action is performed, a dialogue box will appear on the screen.

In the **State name** field, type in the name of the state. The name can not contain any dots (.) or commas (,). Further, the name has to be unique within the set of states in the particular subtopic. A name is required so you can not leave this field blank.



Then select the type of the state by ticking the demanded type in the **type** checkboxes. The *Entry* checkbox is already chosen, since that is the default type.

When a correct name is typed in, click the **Ok** button to create the new state or the **Cancel** button to return to the DMT without creating a state.

Edit state

In order to edit a state you must make select a subtopic and then select **Show states** for that specific subtopic, see section Show states. Then click the **Edit state** button above the **State list**. When this action is performed, a similar dialogue box, as for creating a new state, will appear on the screen, but with the current information of the state inserted in the fields. To edit the state, change the information in the fields in the same way as described in section New state.

Then click the **Ok** button to keep the changes or the **Cancel** button to return to the DMT without changes.

Delete state

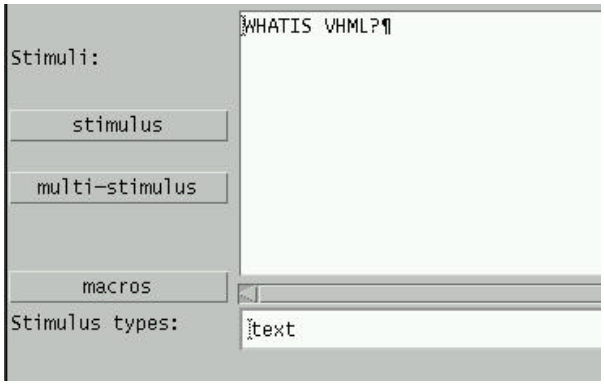
In order to delete a state you must make select a subtopic and then select **Show states** for that specific subtopic, see section Show states. Then click the **Delete state** button above the **State list**. A confirming dialogue box will appear on the screen. If you want to proceed, click the **Ok** button, if not, click the **Cancel** button.

By deleting a state you should be aware of that you also delete all references pointing to that state. Read more about references in the sections Responses, Previous states and Next states.

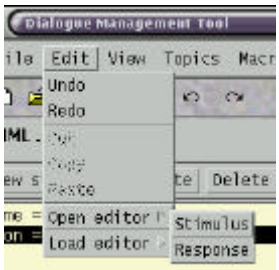
Viewing a state

To view a state, make sure to view the right subtopic by selecting **Show states**, see section show states. This will present the states in the selected subtopic in the **State list**. Then select the state to view in the **State list** by using the mouse or the arrow keys on the keyboard. The information in the specified state will be presented in the fields in the **State information** area. These are the **Stimuli**, **Stimulus types**, **Responses**, **State reference**, **Response weight**, **Previous states**, **Next states**, **Signals**, **Evaluate** and **Other** fields.

Stimuli



The state can have zero or more stimuli. These should be typed into the **Stimuli** field. Use the **stimulus** and **multi-stimulus** buttons on the left hand side of the **Stimuli** field to mark it as zero or more stimuli. The **stimulus** button sets a *stimulus mark* (¶) in the position of the mark, so make certain that the mark is placed after the stimulus. A way to create more than one stimulus at a time is to type in a number of stimuli in the field, one on each row. Then highlight all the stimuli and click the **multi-stimulus** button. In this way, a stimulus mark will be inserted at the end of each row, making each row a separate stimulus.



Macros can be used in order to avoid having too many stimuli. Read more about macros in section Macros.

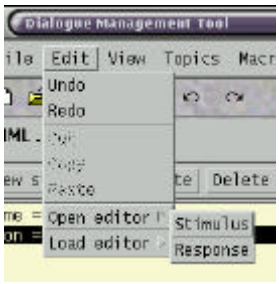
It is also possible to type the stimuli in the editor GVim if that editor is preferred to the user. To open GVim select **Open editor** in the **Edit** menu and then **Stimulus**. Then type the stimuli in the editor. To load the stimuli into DMT select **Load editor** in the **Edit** menu and then **Stimulus**.

When all stimuli have been created, the types of the different stimuli have to be decided. A stimulus can be of several different types depending on the application; *text*, *audio*, *visual* and *haptic*, although *text* is the default value. For example, instead of having "Yes" as a text stimulus, there can be a visual stimulus when the user nods. Since *text* is the default type, it is already specified in the **Stimulus types** field. If the same type is wanted for all stimuli, one type is enough in the field, though every stimulus will get the specified type. If different types are demanded, one type for each stimulus has to be typed in, in the same order as the stimuli.

Responses

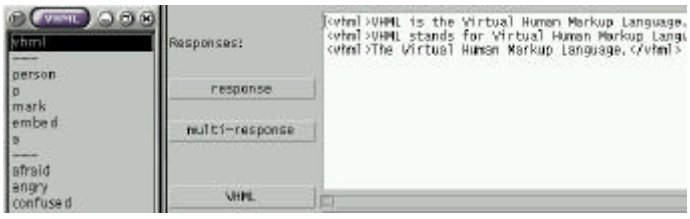
The state can have zero or more responses. A response could be plain text or marked up in any language. For example, the question and answer structure in a FAQ file could be maintained by using the stimuli and responses. The response could also be marked up to direct or control the way in which the response is presented, for example, by using HTML anchors.

In the **Responses** field, type in the different responses. Use the **response** and **multi-response** buttons on the left hand side of the **Responses** field in order to mark it as zero or more responses. The **response** button sets a *response mark* (¶) in the position of the mark, so make certain that the mark is placed after the response. A way to create more than one response at a time is to type in a number of responses in the field, one on each row. Then highlight all the responses and click the **multi-response** button. In this way, a response mark will be inserted in the end of each row, making each row a separate response.



It is also possible to type in the responses in the editor GVim if that editor is preferred to the user. To open GVim select **Open editor** in the **Edit** menu and then **Response**. Then type in the responses in the editor. To load the responses into DMT select **Load editor** in the **Edit** menu and then **Response**.

If the user chooses to type in the responses in the DMT there is support for using the *Virtual Human Markup Language* (VHML)., since VHML can be useful when controlling the output of a TH application. To insert a VHML element into the **Responses** field click the **VHML** button to the left of the field. This opens the **VHML list** with all available VHML elements. To insert a certain VHML element select the element in the **VHML list** by using the mouse or the arrow keys on the keyboard. When the wanted element is marked, insert it to the **Responses** field by double clicking it or using the enter key on the keyboard. Of course, VHML elements can be typed in by hand as any other plain text, but using the **VHML list** prevents misspelling and using of element names that do not exist.



Further, a response has a weight with the default value *0.7*. This can be used by the dialogue manager when there exists more than one response and it has to be decided which one to present. This gives the user a possibility to specify the preferred response to the dialogue manager. For example, a response with a higher weight can be more likely to occur than a response with a low weight. Responses with the same weight

could be used for having a random response. It is up to the dialogue manager to decide which one to use. Also a dialogue manager might be able to change the responses so that when an response is presented to the user the weight decreases so that he same response not appears twice in a row. The default value will automatically appear in the **Response weight** field when inserting responses in the **Responses** field. If the same weight is wanted for all responses, one weight is enough in the field. Every response will get the specified weight. If different weights are wanted, a weight for each response has to be typed in, in the same order as the responses.

State reference

A state can have a state reference instead of responses. This makes it possible for two different states to have the same responses. This is a useful feature when, for example, the user asks a question like "What is VHML?" or if the user previously has been introduced to the concept VHML and asks: "What is that?". These two questions should trigger the same responses, but the first one has to be an entry state and the second a linked state. Though, the first question can be posed during any time in the dialogue and the other question must have a context where "that" refers to something that has been introduced earlier. To avoid having to type in the same responses twice or even more, a state reference may be used. A response that specifies a state reference has exactly the same response as the referred state and hence can not have any additional responses. To specify a state reference, remove all responses, if there exists any, from the **Responses** field. Then type in a state reference in the **State reference** field. The reference should be a *fully qualified name*, i.e. a name that gives the whole search path to a state. For example, a state called *name* in a subtopic *whatis* in a topic *VHML*, has the fully qualified name *VHML.whatis.name.*

Previous states

The state can contain zero or more previous states. The previous states specify the states from which the dialogue could have come.

The previous states are specified in the **Previous states** field. The states referred to must be specified by their *fully qualified names*, i.e. a name that gives the whole search path to a state. For example, a state called *name* in a subtopic *whatis* in a topic *VHML*, has the fully qualified name *VHML.whatis.name.*

Next states

The state can contain zero or more next states. The next states specify into which states the dialogue could move.

The next states are specified in the **Next states** field. The states referred to must be specified by their *fully qualified names*, i.e. a name that gives the whole search path to a state. For example, a state called *name* in a subtopic *whatis* in a topic *VHML*, has the fully qualified name *VHML.whatis.name.*

Signals

The state can contain zero or more signals. A signal enables the match to generate or emit a signal or notification to the dialogue manager, which it may choose to ignore or

handle in some way. For example, if the user says “Good bye”, the dialogue manager may choose to close the connection.

The signals are specified in the **Signals** field. What type of signals there are is up to the dialogue manager to decide, but it should be some predefined value that the dialogue manager know how to handle.

Evaluate

Evaluate can be used for defining a condition that has to be fulfilled before the dialogue is able to move into this particular state. For example, a variable can be set to imply that a state is visited.

The evaluate condition is specified in the **Evaluate** field.

Examples of how to use evaluate can be found on the web at the VHML webpage http://www.vhml.org/documents/DMTL/evaluate.shtml.

Other

The **Other** field can be used for specifying any additional application specific information necessary or simply to add comments about the state.

Help

User manual

A user manual for the DMT can be reached by choosing **Help** in the **Help** menu on the web at http://www.vhml.org/downloads/DMT.

Warning and error messages

Warnings and error messages are presented in the **Error status** field in the bottom of the DMT whenever a forbidden action has been performed. The error messages are also accompanied by a *beep* sound to stress that an error has occurred.

Hints for the user

When using the tool, there is a lot to think about in order to get all the advantages and the best use of the tool.

- Make an outline of the planned overall structure of the dialogue, before starting to implement it. This will often sort out your thoughts and facilitate the constructing of the dialogue.
- Take notice of all warning and error messages. If these are ignored, data may go missing and the dialogue may turn out to be incorrect.
- Use the facility to tear off the list with topics, subtopics and macros when working at a certain location in the dialogue for a longer time. Have the list

placed at the desktop to reduce the number of mouse clicks in the menus and hence make the construction more efficient.

- Begin with only one stimulus in each state. This can later be extended to multiple stimuli or a macro can be constructed, which the stimulus may be translated to.
- Remember that all stimuli that need a known context must be in linked states and should not be merged with the stimuli that can be used independently of the context, which should be placed in an entry state. This may duplicate the state, and in this case the use of state references between those states can be a good solution.
- Make the connections between the states, by using next states or previous states, at the end of the construction. Thus, it is not possible to do a reference to a state that does not exist in the dialogue.
- Be thoughtful when selecting the names of the states, subtopics and topics. It is important that the names are intuitive, especially when typing in references to other states as for previous states, next states and state references.

Appendix F

Test schedule

The testing for DMT is divided in ten parts. There are a number of issues that has to be investigated for each part. These are listed here.

General

- Are all requirements fulfilled?
- Is every action possible to do in any possible order?
- Do all alternatives in the menus work?
- Do all images shortcuts work?
- Are all dialogs correct and does all functions work?

Graphic user interface

- Are the colours good?
- Is everything correct spelled?
- Are all names intuitive and correct?
- Are the objects intuitive?
- Are all objects grouped in an intuitive way?
- Is it obvious what belongs to what?
- Is it clear what is static information and where the user is supposed to fill in data?
- Does it exist image shortcuts for all relevant functions?
- Is the size of the window and all the objects good?

Information presentation

- Is all information presented in a good way?
- Is all information presented at the correct place?
- What happens if the data fills the field?
- Does the scroll work in a good way?
- Is it possible to erase in all text fields?

Topic

- Does the name appear in the show topic/subtopic label when selecting a topic?
- Does a new topic appear in the topic list?

Subtopic

- Does the name appear in the show topic/subtopic label when selecting a subtopic?
- Does a new subtopic appear in the subtopic list?
- Can all the subtopics requirements be fulfilled on any subtopic level?

State

- Is the correct information presented in the state list?
- Is the information updated dynamically in the state list?

Macro

- Does a new macro appear in the macro list and in the list connected to the macro button?
- Do the macros appear in the stimulus in the right way and is the marker set at the right spot after clicking the list?

VHML

- Is the list complete and correct?
- Do the tags appear in the response in the right way and is the marker set at the right spot after clicking in the list?

Error control

- Is the user noticed as soon as a wrong action has been done?
- Are “Warning” versus “Error” at the right time?
- Are all error message correct spelled and formulated?
- Will the user understand how to correct the mistake when getting the message?
- Is the mark set on a suitable spot after getting the message?
- Is everything that is not possible to do shadowed in the menus?
- Are all buttons that are not possible to use shadowed?

Other

- Is the use of the tab key intuitive?
- Is there a good way of getting help?
- Does the transformation to the DOM tree work?
- Does the transformation to the DMTL file work?

Appendix G

The Mystery at West Bay Hospital

A murder has been committed at the West Bay Hospital. John Smith was this Sunday found dead in his bed, obviously choked. His roommate, Paul Windsley, heard some strange noise from the other side of the partition that separates the room into two and rang the alarm at around 3 PM.

John was being treated at the hospital for a ruptured lung after a sad accident. His colleague Amy Goldman has accidentally run him over at the parking area after work.

Visiting hours at the hospital are 12 to 3 PM every day. This particular day John had two visitors, his girlfriend Patricia Stone and his colleague Amy.

Only three people were working at the hospital this day, Dr Goldman, the nurse Alice Duffy and Susan Leonard, who cleans the hospital and also is John's sister.

All people involved are seen as suspects. You are a well know detective who is send after in order to find out who the murder is. You will receive help from Tom Cartier, the policeman who has started up the investigation. He will be able to answer questions regarding circumstances concerning the murder, the suspect's motives, etc. You can also take in any of the suspects for questioning to hear what they can say for their defense.

The six suspects can be found at the top of the screen. Click on the one that you want to question and type your questions, one at a time, in the text field at the bottom of the application. When you think you know who is the murderer, click on the judge to deliver your answer. If you would like to give up or just get the correct answer, simply click the judge and ask for the solution.

Appendix H

VHML Questionnaire

Dear member of the European Union 5th Framework Research,

We are students from Computer Science at Linköping University, Sweden and since the middle of July, we have been doing our degree project at Curtin University of Technology, Perth, Australia. This project is part of the European Union 5th Framework Research and involves verification, validation and evaluation of the Virtual Human Markup Language (VHML). The VHML working draft version 0.3 (www.vhml.org/documents/VHML) is now finished and in order to make the specification even better in the future, we would like to receive opinions from people with your expertise. Remember we are only at level 0.3, so your feedback does not need to be detailed.

During the verification of VHML seven criteria were defined. These were used as the basis for all decisions taken when improving the language. The criteria are:

- *Completeness.* The language must be complete, i.e. it should cover all functionality that should be provided.
- *Simplicity.* The language should aim to be as simple as possible, i.e. not include any ambiguous features. That would keep the language fairly small and surveyable. Though, this should not affect the previous criteria. In order to fulfil this criteria, elements that have the same functionality should be merged.
- *Consistency.* To make it easier for the user to learn the language, it must be consistent, i.e. the syntax should follow a certain pattern. For example, the element names should be in the same form and have the same sorts of attributes.
- *Intuitivity.* If the language is intuitive, the user will not always need to consult the specification to be able to use the language. The names of the elements and attributes should be self-describing and able to tell the user what they can be used for.
- *Abstraction*. By using a high abstraction level, the language will be easier to understand.
- *Usability*. The language should provide features that suit both beginners and advanced users.
- *Standardization.* The language should as far as possible follow existing standards for the different parts of VHML. It is important that the language it builds on is or will become a standard. In case it is probable that it will become a standard it is important to provide features so the language easily can be changed to follow the standard in the future.

We certainly hope that you can spend a few Man Minutes to read through the VHML specification and depending on the areas of your expertise answer the questions in the form. It will approximately take 30 ±10 minutes to read the document plus an extra 30 ±10 minutes to think about it and express your thoughts in the form. We would appreciate it if you return the form to us before 4th of November. The reasons for this short notice is that we would like to increase the usefulness of VHML and at the same time include the evaluation in our project thesis.

Thank you for taking the time in helping us with the development of the new and exciting markup language.

Regards, Emma Wiknertz, Linda Strindlund and Camilla Gustavsson.

My area/areas of expertise is/are (mark the appropriate area with an X):

Image analysis		
Image synthesis		
Speech analysis		
Speech synthesis		
Gestures		
Emotions		
XML		
Standards		
Other		Please specify:

For each question, mark **Yes** or **No** with a cross and use the space after the question to comment your answer. Further, if you find any question hard to answer because of your lack of expertise in that area, just leave the lines blank.

THE VHML DOCUMENT STRUCTURE

The following questions relate to the overall style and content of the VHML document. The sections in the document about BAML, DMML and XHTML has not been given much effort and are therefore not of importance for this evaluation.

1. Is the document complete? Are all sections included or is there something missing?

   Yes: **No:**

2. Do all parts of the document have relevance to the VHML specification?

   Yes: **No:**

3. Is the layout of the document good?

   Yes: **No:**

4. Is the information presented in a logical order?

   Yes: **No:**

5. Is the document clear and easy to understand?

   Yes: **No:**

6. Is there enough information in the document for a programmer to be able to use VHML?

   Yes: **No:**

7. Is there enough information in the document for a programmer to be able to implement VHML?

   Yes: **No:**

8. Is the electronic document easy to use? Do you prefer using an online or a printed document?

   Yes: **No:**

THE VHML SPECIFICATION

The following questions relate to VHML as a language. Use your area of expertise, we would like comments on the follow aspects of VHML. Do not bother commenting the sections in the document about BAML, DMML or XHTML.

Completeness

1. Does the specified functionality cover all your needs? Would you like to add any:
   Sub languages?
   Elements?
   Attributes?

   Yes: **No:**

Simplicity

1. Is it possible to distinguish between all terms and are all terms relevant regarding:
   Sub languages?
   Elements?
   Attributes?

   Yes: **No:**

2. Is the structure of the language simple?

   Yes: **No:**

3. Can any improvements or simplifications be done to the DTD (the DTD can be found as Appendix A in the VHML document.)?

   Yes: **No:**

Consistency

1. Is the language consistent regarding the form of:
   Element names?
   Attribute names?
   Attribute values?

   Yes: **No:**

Intuitivity

1. Are the names of the objects self-describing, so that a programmer would be able to guess the names without consulting the specification?

   Yes: **No:**

2. Is the structure of the language intuitive?

   Yes: **No:**

Abstraction

1. Is the level of abstraction acceptable?

   Yes: **Too low:** **Too high:**

2. Does the DTD reflect the abstraction of the language?

   Yes: **No:**

Usability

1. Does it suit both beginners and advanced users? What improvements can be done in that matter?

   Yes: **No:**

2. Does VHML suit all Virtual Human/Talking Head situations you have considered?

   Yes: **No:**

Standardisation

1. The speech part of VHML follows the current draft of SSML[1] (W3C) and in the case when there is no correspondence, VoiceXML and SML[2] (created by Stallo at Curtin University of Technology) have been considered. Are there any other standards that should be considered for the same part or other parts of VHML?

   Yes: **No:**

GENERAL COMMENTS

1. Do you have any further comments that were not covered in the questions above?

Thank you for your time!

[1] http://www.w3.org/TR/speech-synthesis
[2] http://www.computing.edu.au/~stalloj/project/honours/thesis

Appendix I

Mystery Questionnaire

The Mystery at West Bay Hospital

The purpose of this questionnaire is to get valuable feedback from users of *The Mystery at West Bay Hospital*.

This feedback will be used in the Master Thesis *Verification, Validation and Evaluation of the Virtual Human Markup Language (VHML)* by Gustavsson, Strindlund and Wiknertz. The thesis project was performed at Curtin University of Technology during the 2nd semester 2001.

This feedback will be used in the PhD research *The Design and Effect of Synthetic Character-Agents in Computer Mediated Information Delivery* by Haddad H.

This exercise will take approximately 30 minutes. That includes trying to solve the mystery and fill in the questionnaire.

PLEASE NOTE:

- You do NOT have to take part in this questionnaire.
- If you find any of these questions intrusive, feel free to leave them unanswered.
- Any data collected will remain strictly confidential, and anonymity will be preserved.

If you have any questions, feel free to ask them either during the evaluation or send an email to one of us.

Camilla Gustavsson c.gustavsson@home.se
Hanadi Haddad hanadi77@hotmail.com
Linda Strindlund linda.strindlund@home.se
Emma Wiknertz wiknertz@home.se

THANK YOU

Section 1 – Personal and Background Details

Age: _____ ❑ Female ❑ Male

Is English your first spoken language? ❑ Yes ❑ No

Do you regularly solve mysteries? ❑ Yes ❑ No

Have you ever used a Talking Head application before? ❑ Yes ❑ No

Section 2 – The Mystery at West Bay Hospital

1. What are the full names of the characters presented? Write down as many as you can remember.

2. Briefly describe the physical appearance of each character presented.

3. Use the scale below to indicate the extent you would prefer this kind of character (realistic) to a more cartoon-like character (cartoon).

   a. Put a cross in the space that best expresses your preference.

      Realistic ❑ ❑ ❑ ❑ ❑ ❑ Cartoon

b. Explain why you chose this is your preference.

4. Did you solve the mystery? ❑ Yes ❑ No
If yes go to question 6.

5. Did you give up solving the mystery? ❑ Yes ❑ No
If yes, why?

6. Approximately how much time did you spend with *The Mystery at West Bay Hospital*?_____

7. How many guesses did you make about who the murderer was?

❑ 0 ❑ 1 ❑ 2 ❑ 3

8. Did you ask the judge for hints? If no, go to question number 10.

❑ Yes, once ❑ Yes, several times ❑ No

9. Did the judge's hints help you to solve the mystery?

❑ Yes, totally ❑ Yes, nearly ❑ Yes, a little ❑ Yes & No ❑ No, not at all

10. Did the characters ever say that they did not know the answer to your question? If no, go to question number 13. ❑ Yes ❑ No

11. Did it matter? ❑ Yes ❑ No
    If no, go to question 13.

12. How did you find this?

❑	❑	❑	❑	❑
Not annoying at all	Slightly annoying	Annoying	Very annoying	Terrible

13. Were all the answers relevant according to the posed question? If yes, go to question number 15. ❑ Yes ❑ No
    If no, try to give an example that you remember.

14. Was it possible to reword a question in order to get a satisfactory answer?

    ❑ Yes ❑ No ❑ I did not try

15. How much did you enjoy *The Mystery at West Bay Hospital*?

❑	❑	❑	❑	❑
Very much	Much	Little	Very little	Not at all

    Why/why not?

16. How did you find the complexity of *The Mystery at West Bay Hospital*?

❑	❑	❑	❑	❑
Very simple	Simple	Average	Complicated	Very complicated

Why?

If you have any other comments about *The Mystery at West Bay Hospital*, please write them below.

LINKÖPING UNIVERSITY
ELECTRONIC PRESS

På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Camilla Gustavsson, Linda Strindlund & Emma Wiknertz